ＭＡＴＬＡＢ 5.x版 対応
ＭＡＴＬＡＢ対応ライブラリ
ADSP324-41Ｂ ／ ADSP404-241Ｂ ／ ADSP674-341
ＭＡＴＬＡＢ対応ＵＩＬ
ADSP324-49Ｂ ／ ADSP404-249Ｂ ／ ADSP674-349
モデルコントロール
ADSP324-50 ／ ADSP404-250 ／ ADSP674-350

５．ｘ．ｘ版　インストール・ガイド

中部電機株式会社

＜＜　目　次　＞＞

1　概要 2
2　全体の手順 4
3　インストール手順 5
3.1　ＤＳＰとＤＳＰ用コンパイラーセットアップ 5
3.2　ＭＡＴＬＡＢのインストール 5
3.3　旧版ＭＡＴＬＡＢ対応ライブラリのアンインストール 5
3.4　ＭＡＴＬＡＢ対応ライブラリのインストール 6
4　異なる種類のＤＳＰの同時利用 11
5　5.02.00 版以降にＡＤＳＰ３２４－００Ａを使用する際の注意事項 12

1　概要

本書は、ＭＡＴＬＡＢ対応ライブラリ（マルチ）「ＡＤＳＰ３２４－４１Ｂ／ＡＤＳＰ４０４－２４１Ｂ／ＡＤＳＰ６７４－３４１」及び、ＭＡＴＬＡＢ対応ＵＩＬ「ＡＤＳＰ３２４－４９Ｂ／ＡＤＳＰ４０４－２４９Ｂ／ＡＤＳＰ６７４－３４９」及び、モデルコントロール「ＡＤＳＰ３２４－５０／ＡＤＳＰ４０４－２５０／ＡＤＳＰ６７４－３５０」（共に 5.02.00 版）のインストール方法について説明しています。本書が説明している内容は、ＤＳＰのセットアップを含めた全体の手順概略と、本製品の具体的インストール手順です。ＤＳＰのセットアップ、コンパイラーのセットアップ等については、個々の製品の説明書が必要になりますので、本製品以外に以下の製品の説明書等をお手元に用意してください。

ＡＤＳＰ３２４－４１Ｂ／ＡＤＳＰ３２４－４９Ｂ／ＡＤＳＰ３２４－５０の場合

○ ＤＳＰボード「ＡＤＳＰ３２４－００Ａ スタートアップガイド」
○「ＡＤＳＰ３２４－４１Ｂ ライセンスシート」
○「ＡＤＳＰ３２４－４９Ｂ ライセンスシート」
○「ＡＤＳＰ３２４－５０ ライセンスシート」
○「ＡＤＳＰ３２４－４１Ｂ ユーザーズ・マニュアル」
○「ＡＤＳＰ３２４－４９Ｂ ユーザーズ・マニュアル」
○「ＡＤＳＰ３２４－５０／ＡＤＳＰ４０４－２５０／ＡＤＳＰ６７４－３５０ チュートリアル」
○ ＤＳＰ用コンパイラ ＣＰＬ３２０ＦＬＴ－ＭＳ の説明書
○ ＭＡＴＬＡＢ／ＳＩＭＵＬＩＮＫ／ＲＴＷの説明書

ＡＤＳＰ４０４－２４１Ｂ／ＡＤＳＰ４０４－２４９Ｂ／ＡＤＳＰ４０４－２５０の場合

○ ＤＳＰボード「ＡＤＳＰ４０４－００ スタートアップガイド」
○「ＡＤＳＰ４０４－２４１Ｂ ライセンスシート」
○「ＡＤＳＰ４０４－２４９Ｂ ライセンスシート」
○「ＡＤＳＰ４０４－２５０ ライセンスシート」
○「ＡＤＳＰ４０４－２４１Ｂ ユーザーズ・マニュアル」
○「ＡＤＳＰ４０４－２４９Ｂ ユーザーズ・マニュアル」
○「ＡＤＳＰ３２４－５０／ＡＤＳＰ４０４－２５０／ＡＤＳＰ６７４－３５０ チュートリアル」
○ ＤＳＰ用コンパイラ ＣＰＬ３２０ＦＬＴ－ＭＳ の説明書
○ ＭＡＴＬＡＢ／ＳＩＭＵＬＩＮＫ／ＲＴＷの説明書

ＡＤＳＰ６７４－３４１／ＡＤＳＰ６７４－３４９／ＡＤＳＰ６７４－３５０の場合

○ ＤＳＰボード「ＡＤＳＰ６７４－００ スタートアップガイド」

○「ＡＤＳＰ６７４－３４１ ライセンスシート」

○「ＡＤＳＰ６７４－３４９ ライセンスシート」

○「ＡＤＳＰ６７４－３５０ ライセンスシート」

○「ＡＤＳＰ６７４－３４１ ユーザーズ・マニュアル」

○「ＡＤＳＰ６７４－３４９ ユーザーズ・マニュアル」

○「ＡＤＳＰ３２４－５０／ＡＤＳＰ４０４－２５０／ＡＤＳＰ６７４－３５０ チュートリアル」

○ ＤＳＰ用コンパイラ ＣＰＬ３２０ＦＬＴ－ＭＳ の説明書

○ ＭＡＴＬＡＢ／ＳＩＭＵＬＩＮＫ／ＲＴＷの説明書

2　全体の手順

本製品が使える状態までセットアップする手順（及び参照する説明書）は以下の通りです。

１、ＤＳＰとＤＳＰ用コンパイラーセットアップ（ｽﾀｰﾄｱｯﾌﾟｶﾞｲﾄﾞ）

２、ＭＡＴＬＡＢのインストール（MATLAB の説明書、及び本書）

３、旧版のＭＡＴＬＡＢ対応ライブラリのアンインストール

４、ＡＤＳＰＸＸ４－Ｘ４１Ｂ／Ｘ４９Ｂ／Ｘ５０のインストール（本書）

１～２項については、各資料を参照のうえ、セットアップを行ってください。

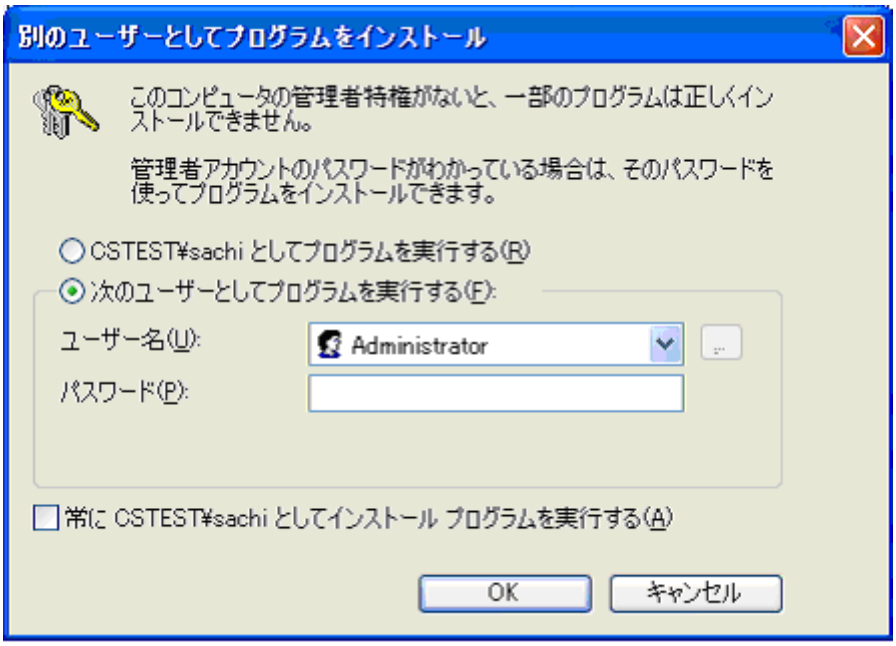
３～４項については、次ページ以後の説明に従ってください。

≪注意≫

インストールは、管理者権限を持つユーザにて実行してください。管理者権限のないユーザにて実行した場合は正常にインストールされませんので、一度アンインストールしてから再度インストールを実行してください。

管理者権限のないユーザにてインストールを開始すると以下のメッセージが表示される場合があります。ユーザの切り替えをして、再度インストールをしてください。

インストール先フォルダは、名称に日本語・空白を含まないフォルダへインストールしてください。例えば、”C:¥program files”の下へインストールするのは NG です。



3　インストール手順

3.1　　ＤＳＰとＤＳＰ用コンパイラーセットアップ

ＤＳＰボードに添付している「スタートアップガイド」及び、ＤＳＰ用コンパイラ（ＣＰＬ３２０ＦＬＴ－ＭＳ／ＣＰＬ３２０６Ｘ－ＭＳ／ＨＬＬ３２０６Ｘ－ＭＳ）の説明書を参照し、ＤＳＰのセットアップとコンパイラーのインストールからＤＳＰボードの動作確認まで行ってください。
既にインストールしてある場合は、再インストールは不要です。

3.2　　ＭＡＴＬＡＢのインストール

使用出来るＭＡＴＬＡＢは、5.x 版です。
既にＭＡＴＬＡＢ5.x 版がインストールしてある場合は、再インストールは不要です。
それ以外の版をお使いの場合や、まだＭＡＴＬＡＢがインストールされていない場合は、ＭＡＴＬＡＢの取扱説明書を参照のうえＭＡＴＬＡＢのインストールを行ってください。

3.3　　旧版ＭＡＴＬＡＢ対応ライブラリのアンインストール

旧版の弊社ＭＡＴＬＡＢ対応ライブラリ「ＡＤＳＰＸＸ４－４１Ｂ／Ｘ４９Ｂ／Ｘ５０」をご利用の場合は、それらの製品をアンインストールしてください。

5.00.00 版より前の場合
アンインストールは、Windows のコントロールパネルにある、「アプリケーションの追加と削除」を起動し、下記の項目を選択し、「追加と削除ボタン」で削除します。

ADSP32X MATLAB 5.x Library	ADSP324 の場合
ADSP404 MATLAB 5.x Library	ADSP404 の場合
ADSP674 MATLAB 5.x Library	ADSP674 の場合

これら全てを削除して下さい。

5.00.00 版以降の場合
アンインストールは、Windows のコントロールパネルにある、「アプリケーションの追加と削除」を起動し、「MATLAB 対応ライブラリ for MATLAB5.x Ver x.xx.xx」の項目を選択し、「追加と削除ボタン」で削除します。

補足：上記の x.xx.xx は MATLAB 対応ライブラリのバージョンを表します。

3.4　　ＭＡＴＬＡＢ対応ライブラリのインストール

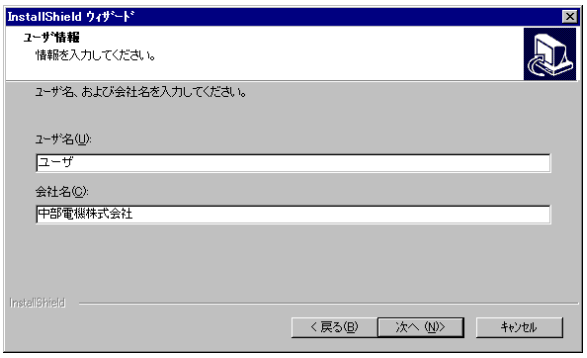
最後に本製品をインストールします。

CD-ROMをCDドライブに挿入するとインストーラが自動的に起動します。起動しない場合は、エクスプローラ等で ＣＤの¥Setup を起動します。

【１】ようこそ

この画面が出たら、次へ(N) ボタンを押してください。

【２】ユーザ情報の入力



ユーザ情報を入力し次へ(N)ボタンを押してください。

【３】購入製品の確認

購入製品を確認し、チェックしてください。

【４】ＭＬＬＩＢのライセンスコードの入力

次に、“ＭＬＬＩＢのライセンスコードの入力”画面が出ます。
ＭＡＴＬＡＢ対応ライブラリ（ADSP324-41B、ADSP404-241B、ADSP674-341）のライセンスシートに記載されているパスワードを入力してください。
パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている － 記号も省略出来ません。
正しく入力したら、次へ(N) ボタンを押してください。

【５】ＵＩＬのパスワードの入力

ＵＩＬを購入された場合のみ、“ＵＩＬのパスワードの入力”画面が出ます。
ＭＡＴＬＡＢ対応ＵＩＬ（ADSP324-49B、ADSP404-249B、ADSP674-349）のライセンスシートに記載されているパスワードを入力してください。
パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている － 記号も省略出来ません。
正しく入力したら、次へ(N) ボタンを押してください。

【６】モデルコントロールのパスワードの入力

モデルコントロールを購入された場合のみ、“モデルコントロールのパスワードの入力”画面が出ます。

ＭＡＴＬＡＢ対応モデルコントロール（ADSP324-50、ADSP404-250、ADSP674-350）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている － 記号も省略出来ません。

正しく入力したら、次へ(N) ボタンを押してください。

【７】ＭＡＴＬＡＢのフォルダーの選択

ＭＡＴＬＡＢがインストールされているディレクトリ（フォルダー）を指定します。

ＭＡＴＬＡＢをインストールしたディレクトリ（フォルダー）が表示されているディレクトリと異なる場合は、参照(R) ボタンを押しそのディレクトリ名を設定してください。

ＭＡＴＬＡＢのディレクトリーが正しければ　次へ(N) ボタンを押します。

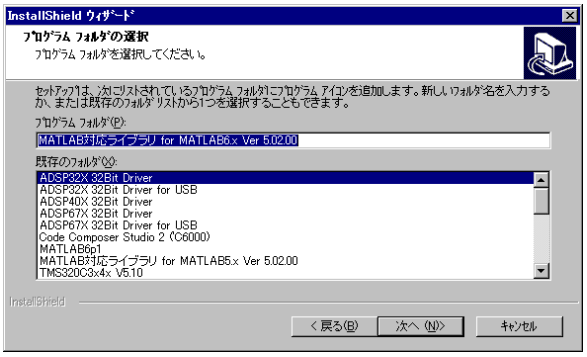
【８】Ｃコンパイラーのフォルダーの選択

続いて、ＤＳＰの開発環境（ＴＩ社製Ｃコンパイラー）をインストールしたフォルダーを指定します。

ＤＳＰ用Ｃコンパイラーをインストールしたフォルダーが表示と異なる場合は、参照(R) ボタンを押し、そのフォルダー名を設定してください。

ＤＳＰの開発環境のフォルダーが正しければ、次へ(N) ボタンを押してください。

【９】プログラムフォルダーの選択



次に、ショートカットを追加するプログラムフォルダーの選択をします。

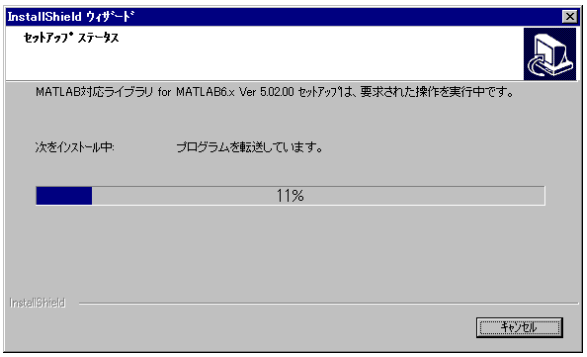
表示されているフォルダー以外のフォルダーにショートカットを追加する場合は、そのフォルダー名を選択又は入力します。

内容を確認したら、次へ(N) ボタンを押してください。

【１０】ファイルコピーの開始

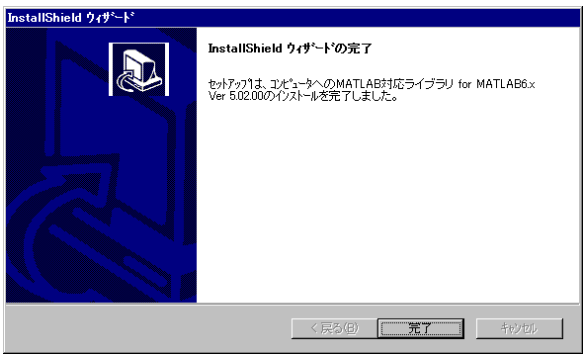
ファイルコピーの開始画面が表示されます。表示内容を確認してください。
内容が誤っているようであれば、戻る(B) ボタンで元に戻って再設定できます。
内容を確認したら、次へ(N) ボタンを押してください。

【１１】セットアップの実行



セットアップステータスが表示され、ファイルのコピーが自動的に行われます。
画面の表示及びプログレスバーがインストール作業の進行と共に変化します。

【１２】ウィザードの完了



最後に、“InstallShield ウィザードの完了“、が表示されます。
完了ボタンを押してください。これでインストールは完了です。

4　異なる種類のＤＳＰの同時利用

ＭＡＴＬＡＢ対応ライブラリ／ＭＡＴＬＡＢ対応ＵＩＬ／モデルコントロールは、以下の表に示す種類のＤＳＰボードに対応しています。

ＤＳＰボード	MATLAB 対応ＬＩＢ	MATLAB 対応ＵＩＬ	モデルコントロール
ADSP324-00A	ADSP324-41B	ADSP324-49B	ADSP324-50
ADSP404-00	ADSP404-241B	ADSP404-249B	ADSP404-250
ADSP674-00	ADSP674-341	ADSP674-349	ADSP674-350

同じ種類のＤＳＰは１つのＭＬＬＩＢ／ＭＬＵＩＬ／モデルコントロールで動作させる事が出来ます。

２種類以上の異なる種類のＤＳＰを使用するは、それぞれのＤＳＰ用のソフトウェアをインストールする必要があります。その場合の注意点について述べます。

複数の製品をインストールする場合以下の注意点が必要です。

◎ＭＡＴＬＡＢを複数それぞれのＤＳＰ用にインストールしてください。１つのＭＡＴＬＡＢ上に２種類以上のＤＳＰ用の環境をインストールする事は出来ません。

◎作業フォルダーを区別してください。同じフォルダーで異なるＤＳＰ用のコードを作成する事は出来ません。

◎ブロック線図の共用は可能です。同じ版のＭＡＴＬＡＢ間であれば、例えば ADSP324-00A 用に作成したブロック線図を、ADSP674-00 用のフォルダーにコピーし、そこで ADSP674-00 用のコードを作成する事は可能です。

◎コンパイラの共用は　ADSP324-00A と ADSP404-00 との間では同じコンパイラを共用できます。コンパイラは個別にインストールする必要はありません。

5　5.02.00 版以降にＡＤＳＰ３２４－００Ａを使用する際の注意事項

ＡＤＳＰ３２４－００Ａの場合にリアルタイムモニタ、ＵＩＬ、モデルコントロールで使用するＤＬＬ（ダイナミック・リンク・ライブラリ）はＵＳＢ接続との互換性を重視し、ＩＳＡ接続での**パソコン－ＤＳＰ間の通信速度が低下**しています。ＤＳＰの動作速度には関係ありませんが、過去にＩＳＡ接続で使用したパソコン側（ＲＴＷでのモニタ、ＵＩＬを使用したＶＢプログラムなど）の動作速度が間に合わない場合はＩＳＡ専用ＤＬＬ（前バージョンのもの）を使用して下さい。今後、新規に作成する場合は共用ＤＬＬを使用して下さい。切り替え方法は スタート～プログラム～ＭＡＴＬＡＢ対応ライブラリ～ＤＬＬ切替～「ＩＳＡ専用ＤＬＬへ切り替え」 をクリックして下さい。共用ＤＬＬへ戻す場合は上記同様～「共用ＤＬＬへ切り替え（標準)」 をクリックして下さい。

ＵＳＢ接続にするには別売のＵＳＢ接続ボード ADSP324-146 が必要です。
この作業は WindowsNT/2000/XP の場合は Administrator 権限で行って下さい
この問題はスタンドアローン時には関係ありません。
ＡＤＳＰ４０４－００はＵＳＢ接続には未対応です。
ＡＤＳＰ６７４－００は当初より互換性があります。

接続に関する詳細は ADSP324-00A スタートアップガイド、ADSP674-00 スタートアップガイドを参照して下さい。
最新版のマニュアル、サポートソフトはホームページより無料でダウンロードできます。
中部電機ホームページＵＲＬ(http://www.chubu-el.co.jp)

6　ＭＡＴＬＡＢ対応ライブラリ実行環境について

　コンパイラのバージョンによって、コンパイラの実行に必要な以下の環境変数の設定が動的に実行するようになっています。この設定をする実行ファイルは”コンパイラのインストールフォルダ¥dosrun.bat”にあります。ＭＡＴＬＡＢを起動するにあたって必要に応じて先にこのＢＡＴファイルを実行し環境設定をしてください。

サンプル)

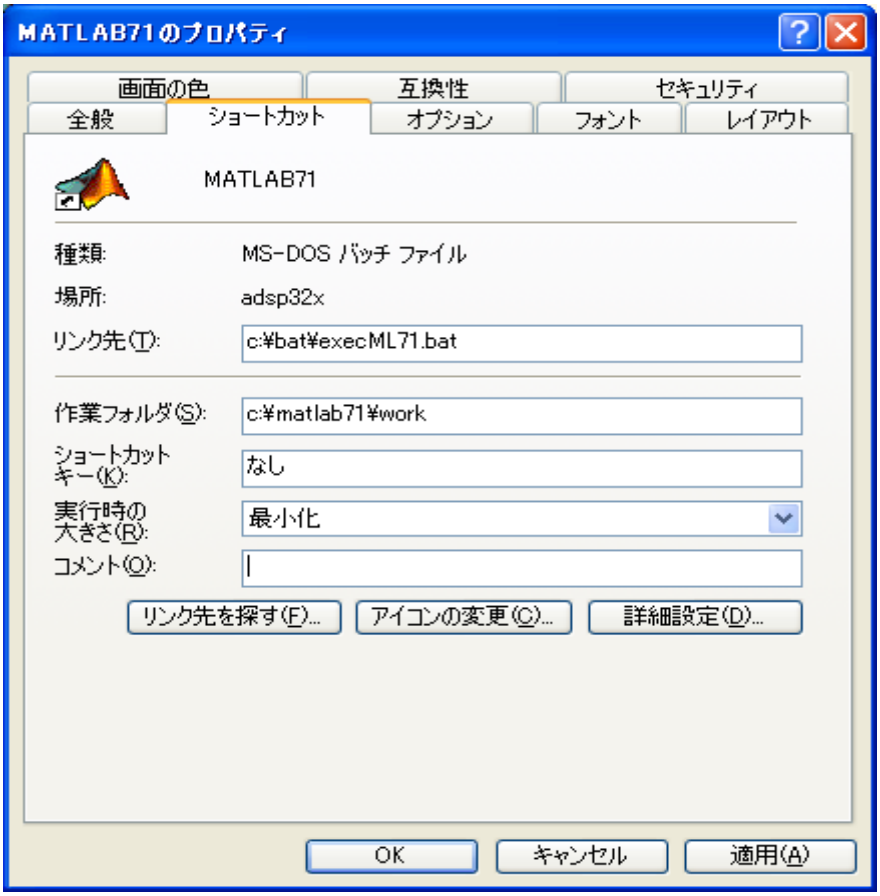
① 起動用ＢＡＴファイルを作成する。

ＢＡＴファイル内容(コンパイラがc:¥CCStudio_v3.1,ＭＡＴＬＡＢがc:¥matlab71にインストールされている時)・・・

```
call c:¥CCStudio_v3.1¥dosrun.bat
call c:¥matlab71¥bin¥win32¥matlab.exe
exit
```

② ショートカットにより①で作成したＢＡＴファイルを起動する。

ＢＡＴファイルが　c:¥bat¥execML71.bat の場合



○MATLAB, SIMULINK, Real-Time Workshop, は米国 The Math Works Inc. の製品です。
○コンパイラー CPL320FLT-MS は Texas Instruments 社の製品です。

◎本書の内容は著作権法により保護されています。本書の内容を無断で複製・転載する事は出来ません。
◎本書及び本製品の内容は、製品の向上等の為予告なく変更される場合があります。悪しからずご了承頂けますようお願いいたします。
◎本書・本製品に関するご質問・ご要望は下記までお願いいたします。
◎落丁・乱丁はお取り替えいたします。

MATLAB 5.x 版 対応
ＭＡＴＬＡＢ対応ﾗｲﾌﾞﾗﾘ
ＭＡＴＬＡＢ対応ＵＩＬ
モデルコントロール
５．ｘ．ｘ版 インストールガイド

第 1 版　　2001.03.21
第 11 版　　2006.10.24

中部電機株式会社 技術部 ＣＳＧ
TEL　　0532-61-9566
FAX　　0532-63-1081
E-mail　csg@chubu-el.co.jp
URL　　http://www.chubu-el.co.jp

ＲＴモデル自動生成

＜＜ 目次 ＞＞

1　概要 3

2　使用上の注意事項 3

2.1　バージョンアップ後のコード生成 3

2.2　モデル名称の制約 3

2.3　ＤＳＰ用コンパイラー版番号 4

3　ＳＩＭＵＬＩＮＫブロックの制約 5

3.1　simulink Sources ～ Pulse Generator 5

3.2　simulink Sources ～ Signal Generator 5

3.3　simulink Sources ～ From File 5

3.4　simulink Sources ～ From WorkSpace 5

3.5　simulink Sinks ～ 5

3.6　simulink Nonlinear ～ MATLAB Fcn 6

3.7　simulink Nonlinear ～ Algebratic Constraint 6

3.8　simulink Nonlinear ～ Fcn 6

3.9　Simulink Extras ～ Additional Discrete ～ Discrete Zero-Pole 7

3.10　Simulink Extras ～ Flip Flops ～ D Flip Flop 7

4　MATLAB 5.x での作成手順 8

4.1　作成に必要なファイル 8

4.2　オプションの設定 8

4.3　実行形式の選択 11

4.4　作成されるファイル 13

4.5　実行手順（リアルタイムモニタの場合） 14

4.6　実行手順（自動実行の場合） 14

5　オプションライブラリ対応 15

5.1.1　ＤＳＰブロックセットの対応 15

5.1.2　それ以外のブロックセットの対応 16

5.2　ライブラリ作成後の作業 17

6　ステップサイズについて 18

6.1　SIMULINK との関係 18

6.2　ステップサイズのデフォルト 18

7　速度向上について 19
7.1　実行速度の相違 19
7.2　高速化するには 19
7.3　ＭＡＰ変更ツール 19
7.4　メモリー不足エラー例 22

1　概要

ＭＡＴＬＡＢのツールボックスのひとつ、Real-Time Workshop を用い、Ｃコードを生成し、ＤＳＰボード上で実行することができます。また、本製品のリアルタイムモニタ又を使用すれば、ただちにＤＳＰ上で実行出来、さらに、その結果をファイルに残すことができます。

ここでは、SIMULINK 上でのシミュレーションのモデル作成は完了しているものとし、これを元にリアルタイムモデルを生成する手順について説明します。

2　使用上の注意事項

2.1　バージョンアップ後のコード生成

本製品のバージョンアップを行った場合、SIMULINK ブロック線図からＲＴモデルを作成する際に、一旦、モデルと同じフォルダーにあるオブジェクトファイルを全て削除してからコード生成を行ってください。

2.2　モデル名称の制約

ＲＴモデルの名称（ブロック線図のファイル名）には、以下に示す制約があります。モデル銘々の際には、充分注意をしてください。

- 名称の先頭は、英字で始めてください。

  良い例：　a3bc.mdl

  悪い例：　3abc.mdl　（理由＝数字で始まっている）

- 名称の文字数は、８文字以内としてください。

  良い例：goodname.mdl

  悪い例：nameislonger.mdl

- 名称で使う文字は英数字と　__　記号のみとしてください。

  良い例：a32_abc.mdl

  悪い例：a(#1).mdl　（理由＝かっこ、#等の記号を使っている）

- 英字の大文字と小文字は同じものとして取り扱われます。

2.3　ＤＳＰ用コンパイラー版番号

使用出来るＤＳＰ用コンパイラーは以下の通りです。

◎ CPL3206X-MS 1.1 版(Code Composer Studio 対応版)

◎ CCS3206X-MS 1.1 版

3　ＳＩＭＵＬＩＮＫブロックの制約

SIMULINK 標準のブロックライブラリの一部は、ＲＴモデルに於いて利用出来ないか、又は利用上の注意が必要です。この章ではこれらについて解説しています。

3.1　simulink Sources ～ Pulse Generator

3.2　simulink Sources ～ Signal Generator

Pulse Generator／Signal Generator は、ＤＳＰが取り扱う時刻変数の精度の関係で、実際の時間（又は SIMULINK で実行した仮想時刻）より、１ステップサイズ分、前後する場合があります。

ＤＳＰは変数を全て単精度実数（32Bit＝8E24）で取り扱います。この精度は、１０進数換算で有効桁数７桁程度です。例えば 0.003 はＤＳＰでは、0.00299999979324 となります。この誤差は避けられません。

3.3　simulink Sources ～ From File

ＲＴＷにてコード生成をした時点のファイルの内容が、生成されるＣソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、ＲＴモデルの実行時点のファイル内容とは無関係となります。

ファイル内容を変更し、その結果をＲＴモデルに反映させるには、再度ＲＴＷでコード生成が必要です。

データが実行コードに組み込まれる関係上、その長さは、ＤＳＰのメモリー容量を超える事は出来ません。

3.4　simulink Sources ～ From WorkSpace

ＲＴＷにてコード生成をした時点のワークスペース変数の内容が、生成されるＣソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、ＲＴモデルの実行時点のワークスペース内容とは無関係となります。

ワークスペース内容を変更し、その結果をＲＴモデルに反映させるには、再度ＲＴＷでコード生成が必要です。

データが実行コードに組み込まれる関係上、その長さは、ＤＳＰのメモリー容量を超える事は出来ません。

3.5　simulink Sinks ～

Sinks グループの Stop Simulation を除く他のブロック（Scope 等）は、ＲＴＷでは利用できません。（MATLAB4.2c では Stop Simulation に加え Scope と ToWorkSpace は利用可能）。

Scope はＲＴＭＯＮの変数モニター機構を使って代用が可能です。具体的には、Scope へ

接続されている信号の出力元のブロック名称を指標して変数表示を行います。

3.6　simulink Nonlinear ～ MATLAB Fcn

ＲＴＷでは利用出来ません。sin 、 cos 等、一部の関数は、Fcn ブロックで代用が可能です。

3.7　simulink Nonlinear ～ Algebratic Constraint

ＲＴＷでは利用出来ません。

3.8　simulink Nonlinear ～ Fcn

利用は可能ですが、式の定義に注意が必要です。

（１）Ｃ言語の標準ランタイムライブラリでサポートされていない関数は利用出来ません。
利用出来る関数／演算子は、以下の通りです。

利用出来る関数	Ｃ言語展開
sin	sin
cos	cos
tan	tan
asin	asin
acos	acos
atan	atan
atan2	atan2
sinh	sinh
cosh	cosh
tanh	tanh
pow	pow
exp	exp
log	log
log10	log10
sqrt	sqrt
floor	floor
ceil	ceil
abs	fabs
mod	fmod
rem	fmod
sgn	sgn （注１）
hypot	hypot

利用出来る演算子	Ｃ言語展開
>	>
<	<
= =	= =
! =	! =
>=	>=
<=	<=
&&	&&
||	||
( )	( )
+	+
-	-
/	/
*	*
!	!

注１：標準ランタイムには含まれませんが、ＲＴＷ用拡張ライブラリが対応しています。

（２）式中に定数を記述する場合で、分数の分子・分母に整数を置く場合は、整数であっても少数形式で記述してください。例えば、１／２を記述する場合は、
1／2.0 又は、1.0／2、又は、1.0／2.0 等としてください。
ＲＴＷではＦｃｎブロックの Expresion に記述された式を、殆どそのままＣ言語ソ

ースに展開します。よって単に 1／2 と記述すると、C 言語の文法規約上、整数計算が行われ、１／２は少数以下切り捨られ 本来０．５が欲しい所が、＝０となってしまい、計算結果に重大な誤りを招きます。

3.9 Simulink Extras ～ Additional Discrete ～ Discrete Zero-Pole

Discrete Zero-Pole (with initial states)は、Simulink の純正ブロック自体が Bug を孕んでいます。以下の手順で対策してください。

（１）MATLAB Command wondiw から Simulink を開く。
（２）Blocksets & Toolboxes をマウス左ダブルクリックし、
Library: Blocksets_and_Toolboxes を開く。
（３）同 Simulink Extras をマウス左ダブルクリックし、Library: Simulink_Extras を開く。
（４）同 Additional Discrete ををマウス左ダブルクリックし、
Library: Simulink_Extras／Additional Discrete を開く
（５）同 Discrete Zero-Pole (with initial states) をマウスで選択する。
（６）メニューから Edit ～ Unlock Library を選択する。
（７）さらに、Edit ～ Look Under Mask を選択し、Library: Simulink_Extras／Additional Discrete／Discrete Zero-Pole (with initial states) を開く。
（８）同、Discrete State Space をダブルクリックし、Discrete State Space ダイアログを開く。
（９）エディットボックス D:の内容が Dd となっているので、これを D に変更する。
（１０）同ダイアログの Close ボタンを押す。
（１１）先の（７）で開いたウィンドウを閉じる。
（１２）先の（４）で開いたウィンドウを閉じる。
（１３）（３）で開いた Library: Simulink_Extras のメニューから、File ～ Save で変更結果を保存する。

3.10 Simulink Extras ～ Flip Flops ～ D Flip Flop

D 入力=1 の状態で、CLK の立ち上がりと、！CLR＝0 が同時に発生する状況では、SIMULINK の実行結果と ＲＴモデルの実行結果に相違が現れます。
SIMULINK では CLK が優先され Q=１となり、RT モデルでは、!CLR が優先され Q=0 となります。

4　MATLAB 5.x での作成手順

4.1　作成に必要なファイル

リアルタイムモデルの作成に必要なファイルは、リアルタイムモデルを記述したブロック線図（モデルファイル＝EXAMPLE.MDL）です。（ブロック線図をファイルに保存すれば自動的にMDL ファイルとして保存されます。）

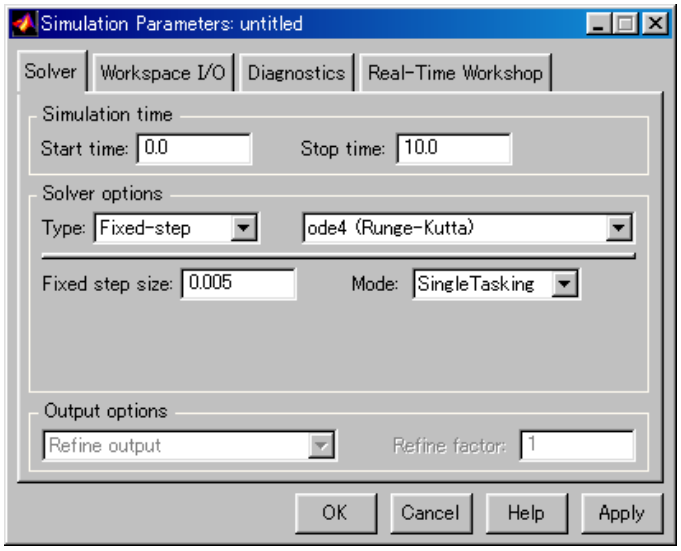
4.2　オプションの設定

モデルのブロック線図を開き、このウィンドウのメニューバー　Tools の RTW Options をマウス左ボタンでクリックし、選択します。すると、リアルタイムオプションウィンドウが開きます。（Simulation メニューの Parameters で開いたものと同じダイアログです)。リアルタイムモデルのＤＳＰ内での実行条件（実行時可変な一部パラメータを除く）の設定はここで行ないます。

Simulation parameters: ダイアログの Solver タブをマウス左クリックします。
Solver Type を Fixed step に変更します。これはリアルタイムモデルでは一定周期での計算しか行えない事を意味しています。
続いて、その右のリストボックスの中から積分則を選択します。モデルが連続系の状態変数を持たない場合は 、discrete (non continuous states) を選択、それ以外の場合はリアルタイムモデルでデフォルトとして利用したいと思う積分則を選択します。

さらにすぐ下の Fixed step size: にステップサイズを設定します。図 5.1.1-1　ではステップサイズが 0.5mSecに設定されています。ここで設定した積分則とステップサイズは、リアルタイムモニターでその設定をデフォルトとして取り出す事が出来ます。
Mode:は Single Tasking を選択してください。

図 5.1.1-1　積分アルゴリズム等設定後



次に、RTW タブをマウス左クリックし、以下を確認します。

System target file:　　chubu.tlc

Template make file:　　a674.tmf　　(メモリモデル Large 時 a674b.tmf)

Make command:　　mk674rtw　　(メモリモデル Large 時 mk674rtwb)

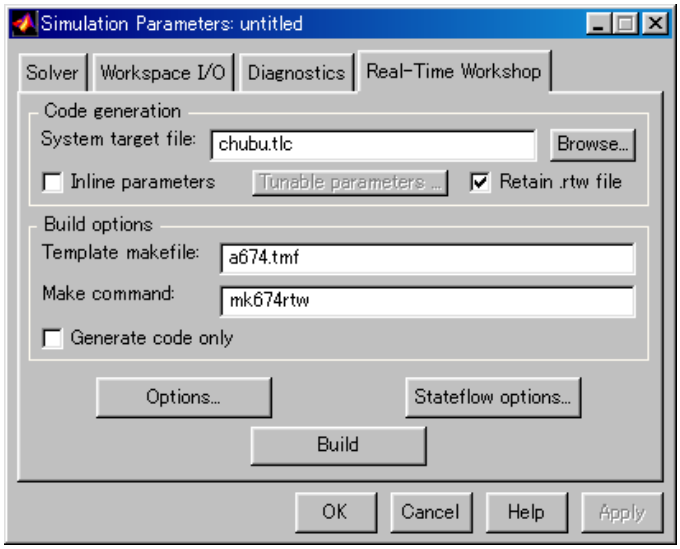
デフォルトは Matlab 対応ライブラリ Ver5.2.0 以降は Large モデルです。

図 5.1.1-2 は、これらの設定が終わった例です。

この設定は、リアルタイムモデル用のタイマードリブンモデルを生成する場合の設定です。

それ以外の実行形式モデルの生成方法は 4.3 実行形式の選択 （11p） を参照してください。

図 5.1.1-2　ＲＴＷオプション（設定後）



Retain .rtw file チェックボックスにチェックマークがついていますが、これは有っても無くても構いません。チェックされていると、モデルと同じフォルダに *MODEL*.rtw が残されます。
Apply ボタンを押して設定を確定してください。出来ればこの段階で、上記設定を保存する意味で、一旦モデルを保存する事をお勧めします。

コード生成は、同ダイアログの Build ボタンを押してください。
ＭＡＴＬＡＢコマンドウィンドウに、 ### で始まるメッセージが何行か表示されます。（一部###以外で始まる行が含まれる場合がありますが、これは異常ではありません。）
これらの行は全てＲＴW関連のメッセージです。コード生成に続きコンパイル・リンク作業が行われ、以下が表示されればコード生成は完了です。
Successful completion of RTW build procedure for model: XXXXX

XXXXX は作成するモデルの名称です。

注）新規以外で作成されたモデルをコード化する場合、弊社ホームページの Q&A「ＭＡＴＬＡＢ対応ライブラリでよくあるＱ＆Ａ」を参照してください。

4.3　実行形式の選択

ＲＴＷによるＤＳＰ用リアルタイムモデルは、その実行方法とモデル計算のきっかけとなるイベントが選択可能です。

実行方法は、リアルタイムモニター又はＵＩＬによる方法“パラメータ可変モデル”と、自動的に実行開始する“スタンドアロンモデル”が可能です。

パラメータ可変モデル は、ステップサイズ、積分アルゴリズム、ゲインブロック等のパラメータ等が実行時に可変で、データロギング、リアルタイムなモニター等が行えます。未完成のモデルを評価したり、パラメータを種々変更してテストする場合等に有用です。

スタンドアロンモデル は、パラメータ等はＲＴＷにてコード化される時点の設定で固定され、実行時には可変出来ない代わりに、プログラムをＤＳＰにロードするとただちに実行が開始されます。これを応用し、ＲＯＭ化（オプションハードウェアが必要）する事により、電源投入と同時に自動的に動作を開始するスタンドアロンモデルを作る事ができます。

実行方法別モデル機能一覧

名称	自動実行	ROM 化	ロギング	ﾊﾟﾗﾒｰﾀ変更	モニター
パラメータ可変	×	×	○	○	○
スタンドアロン	○	○	×	×	×

モデル計算のきっかけとなるイベントは、タイマーイベント、外部割り込みイベントが選択出来ます。

タイマーイベント は、最も一般的で、ＤＳＰのタイマーを用いて一定周期のタイマーイベントを発生させ、これにより、ブロック線図の計算を行います。この周期がＲＴＷによるコード生成時の Step size に相当します。

外部割り込みイベント は、ブロック線図の計算を外部からの割り込みイベントをきっかけとして行います。これにより、外部ＣＰＵや外部ハードウェアに同期したモデル計算が可能となります。SIMULINKの可変ステップサイズと似た動作をしますが、SIMULINKの場合は時間刻みの増減の基準が状態変数の微係数の大小であるのに対し、イベントドリブンは時間刻みが“外部割り込みイベントの発生”に完全に依存します。

実行方法とイベントは、コード生成の際に ＲＴＷ Option ダイアログへの設定で切り替えます。

実行形式・イベント切り替え一覧

イベント	実行方法	上段：Template make file 下段：Make command
タイマー	パラメータ可変	a674.tmf mk674rtw
	スタンドアロン	a674.tmf mk674rtw　ALONE=ON
外部割り込み	パラメータ可変	a674.tmf mk674rtw　EVENTD=ON
	スタンドアロン	a674.tmf mk674rtw　EVENTD=ON　ALONE=ON

☆ Large モデル時 a674.tmf → a674b.tmf

mk674rtw → mk674rtwb を使用してください。

デフォルトは、Matlab 対応ライブラリ Ver5.2.0 以降は Large モデルです。

4.4　作成されるファイル

Real-Time Workshopで作成作業を行なうと、カレントフォルダーに、表 5.1.1-1 に示すファイルが作成されます。作成作業を行なう前のカレントフォルダーに、これらと同名のファイルがあると、上書きにより以前の内容は失われてしまいます。カレントフォルダーにユーザのファイルを置く場合には注意が必要です。

作業は順を追って自動的に行なわれます。最終的に実行に必要となるファイルは、EXAMPLE.OUT です。

表 5.1.1-1　自動生成されるファイル一覧表

番号	生成ファイル名	作成主	備考
1	EXAMPLE.C EXAMPLE.H EXAMPLE.REG EXAMPLE.RTW	Real-Time Workshop	主となるソースファイル
2	EXAMPLE.MK EXAMPLE.LNK EXAMPLE.BAT EXSAMPLE.OPT EXSAMPLE.ITB	Buil command	実行ファイルの作成に 必要なファイル
3	*.OBJ	CL30.EXE	オブジェクトファイル
4	EXAMPLE.OUT EXAMPLE.MAP	LNK30.EXE	実行形式ファイル 番地情報ファイル

注：EXAMPLE は、ユーザが命名したモデルの名称です。

4.5　実行手順（リアルタイムモニタの場合）

リアルタイムモニタの場合は、Windows のプログラムマネージャで、リアルタイムモニタを起動し、これによりロード・実行等を行ないます。この方法は画面上でパラメータや、起動停止等をマニュアリックに操作するので、自動起動等は行なえませんが、最も柔軟性に富んだ方です。
詳細は、リアルタイムモニタの章を参照してください。

4.6　実行手順（自動実行の場合）

自動実行タイプの場合は、以下に示す２つのケースについて説明します。
１、リアルタイムモニターを用いてサブＤＳＰにロードする場合。
２、ＲＯＭ化する場合。
何れの場合も、リアルタイムモデルはスタンドアロンモデルとして作成されていなければいけません。

リアルタイムモニターを用いてサブＤＳＰにロードする場合 は、コード生成作業は全て完了していますので、後はリアルタイムモニターの“サブＤＳＰ”機能でリアルタイムモデルの実行ファイルを指定します。

ＲＯＭ化する場合 は、ＤＳＰ付属ソフトウェアの付属コマンド **UTY674** を用いてＲＯＭ化します。詳しくは ＤＳＰボードのソフトウェアー・ユーザーズ・マニュアルを参照してください。

5　オプションライブラリ対応

ここでは、ＳＩＭＵＬＩＮＫ用のオプションライブラリの対応方法を説明します。

5.1.1　　ＤＳＰブロックセットの対応

ＭＡＴＬＡＢ５．Ｘに於いて、SIMULINK オプションのブロックセットである、ＤＳＰブロックセットの対応方法を説明いたします。以下の作業を行ってください。

１、エクスプローラ等にて、作業用フォルダーを作成します。このフォルダー内にはファイルが何も無い状態としてください。

２、このフォルダーに以下のファイルをコピーします。

C:¥MATLAB¥toolbox¥dspblks¥dspmex¥*.c

３、コピーしたファイルのうち以下ファイルを削除します。これはＲＴＷで作成した実時間モデルではこれらのブロックは対応していない事によります。（注１）

scmtxwk.c（COMPLEX MATRIX To-Workspace）

scplxwks.c（CPMPLEX To-Workspace）

smtrxwks.c（Real Matrix To-Workspace）

４、ＭＡＴＬＡＢ起動し、ＣＤコマンドでカレントディレクトリを作業フォルダーに移動します。

５、以下のコマンドを実行します。

genlib674　dspblks　-r　-q

(Large モデルで作成時は genlib674b にて実行してください。)

６、C:¥MATALB¥rtw¥c¥adsp32x¥lib674¥dspblks.lib が作成されている事を確認したうえで、作業用ディレクトリは削除します。

(Large モデルで作成時 C:¥MATALB¥rtw¥c¥adsp32x¥lib674¥big¥dspblks.lib に作成されます。このライブラリをリンク時は RTW オプションも Large モデルに変更してください。)

以上で作業は終わりです。

注１：同ファイルはコンパイル時にエラーとなります。エラーとなるのは、このブロックがＲＴＷによる実時間モデルでは利用出来ない旨を警告（mexWarnMsgTxt()関数の実行）する目的で記述されている部分です。当該行をコメントアウトすればコンパイルは正常に行われますが、実行が期待通り実行される訳ではありません。この様にして作成したブロックは実時間モデルでは何もしないダミーのブロックとして振る舞います。但し、その旨を理解していれば、コード生成時に当該ブロックを付けたままコード化が可能です。

5.1.2　　それ以外のブロックセットの対応

ＤＳＰブロックセット以外のブロックセットの場合は、弊社標準には対応していません。しかし以下の条件が満たされていれば、殆ど対応が可能と思われます。

対応可能な条件

◎ ブロックが S-Function で記述されていて、その実体はＤＬＬで供給されている事。

◎ S-Function の実体を記述したＣ言語のソースコードが供給されている事。

対応の概略手順は、以下の通りです。

（1）使用したいブロックを、マスクの解除等により、内部を解析し、使用している S-Function の名称を探します。

（2）通常、この S-Function の実体を記述したソースファイルは、S-Function と同じ名称ですから、それを便りにソースファイル（拡張子は.C）を探します。

（3）通常、Ｃ言語ソースファイルは一個所にインストールされていますので、１つが見つかればそれ以外のブロックのソースファイルも同じフォルダーに格納されています。それらをまとめてライブラリ化します。この作業は“ＤＳＰブロックセット”の対応方法又は、“ユーザ定義ブロック”の説明が参考となります。

（4）ライブラリは、出来ればブロックセット毎に一見してそれと判る名称のライブラリとすると良いでしょう。先のＤＳＰブロックセットの場合は、toolbox 下のフォルダー名称 DSPBLKS をライブラリの名称として用いています。

（5）ソースファイルの中で、コンパイルが正常に行われないブロックは、“ユーザ定義ブロック”の説明を参考にして、完全に解決をするのが望ましいのですが、もしそれが、当面使用しないブロックである事が判っていれば、そのソースファイルは除外（ライブラリ作成時の作業用フォルダーからソースファイルを削除）し、必要最小限の S-Function に限定してライブラリ化する方法もあります。

5.2　ライブラリ作成後の作業

ライブラリ作成後は、Real-Time Workshop にて再度コード生成を行ってください。
ＤＳＰのライブラリはスタティックなリンク方式ですので、リンク作業（コード生成作業に含まれる）を行って初めて利用可能となります。

6　ステップサイズについて

6.1　SIMULINK との関係

ＲＴWによるＲＴモデルを生成する場合、実時間モデルでは、可変時間刻みは実現出来ません。実時間モデルでは、過去の時刻にさかのぼって計算をする事が出来ないからです。

可変時間刻み（Variable-Step）みは、微係数を監視して、傾きが小さい場合は時間刻みを粗くして計算省略により時間を短縮すると共に、傾きが大きい場合は時間刻みを細かくして計算精度を維持する意図を併せ持っています。微係数の傾きが大きい場合は自動的に時間刻みが細かくなりますが、固定刻みの場合はこれが行われない為、誤差が増大する可能性があります。

従って、最終的に実時間モデルを作成する予定がある場合、SIMULINK での評価の段階で、固定刻み（Solver Type を Fixed-Step）にして評価する必要があります。

6.2　ステップサイズのデフォルト

ＲＴMＯＮでは、ＲＴWで生成したＲＴモデルのデフォルトステップサイズを取り出す機能を持っています。この機能で取り出せる情報は、モデルの内容により、若干異なります。

この機能で取り出せる情報と、SIMULINK（ＲＴWとの関係）は、以下の表の様になっています。

モデルの内容	RTMON のﾃﾞﾌｫﾙﾄ
連続時間系のブロックのみ	RTW の Simulation parameter の Fixed-step
連続時間系の離散時間系ブロック	一番小さいｻﾝﾌﾟﾙﾀｲﾑ
離散時間系のブロックのみ	（同上）

7　速度向上について

7.1　実行速度の相違

ＤＳＰボードADSP674-00では、使用しているDSPチップの特性上、プログラムの配置により、実行速度が異なります。ボード上に、ＳＢＳＲＡＭ、ＳＤＲＡＭと、チップ内部にオンチップＲＡＭがあります。最も早いのはオンチップＲＡＭですが、チップ上に実装されているオンチップＲＡＭは容量に限りがあります。オンチップＲＡＭより外部のＳＢＳＲＡＭの方が容量が大きいので、大きなプログラムは外部に置きます。

7.2　高速化するには

本ソフトでは、ユーザが作るプログラム（ブロック線図）の大きさが判らない為、標準では全て外部のＳＢＳＲＡＭに実装します。この状態で、希望とする速度が満たせられれば良いのですが、満たせられない場合、プログラム等の配置をオンチップＲＡＭに変更します。

配置を変えるには以下の手順を踏みます。

１、ＭＡＰ変更ツールを使い、配置指定を変更します。

２、ＲＴＷでコード生成をします。

３、メモリーが不足する旨のエラーが出たら１へ戻り、内部に割り付けた領域を外部に戻します。

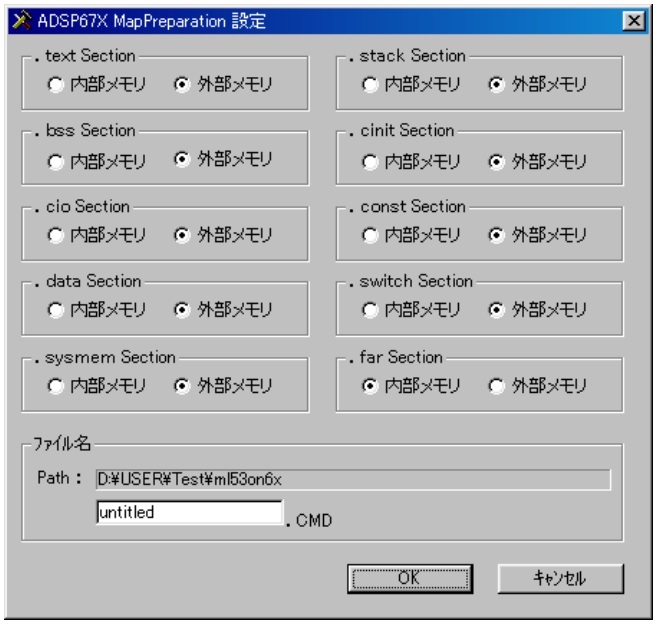
４、エラーが出なければ完了です。

7.3　ＭＡＰ変更ツール

ＭＡＰ変更ツールは、Ｉ／Ｏブロックライブラリのブロック線図に有る、“ＭＡＰエディットボタン”（下図）を目的とするブロック線図に貼りつけ、これをダブルクリックします。

DSP Memory Map

次のような画面が開きます。



本ソフトではＤＳＰに実装されるユーザプログラムを、それを構成する要素（セクション）毎に分けて、そのセクション毎に配置する場所をチップの内部か外部か選択できる様になっています。

.text Section 等のグループは、個々のセクションを表しています。

ファイル名のグループは、これらの情報を保存するフォルダーとファイル名を表しています。この名称は、ＭＡＰ変更ツールをブロック線図から呼び出した場合は変更してはいけません。又、このツールにより自動生成されたファイル（上の例では untitled.cmd）は、直接変更してはいけません。

配置の変更は、各セクションのグループのラジオボタンで、内部メモリーか外部メモリーかを選択し、［ＯＫ］ボタンを押します。設定された内容が書き込まれます。

変更後の内容は次回このツール起動時に表示されます。現在の配置を確認する目的でこのツールを起動した場合は、［キャンセル］ボタンを押して下さい。

設定（又は表示）が終わったらＭＡＴ変更ツールを終了してください。ＭＡＰ変更ツールを起動したままでは、ＭＡＴＬＡＢ／ＳＩＭＵＬＩＮＫの操作は出来ません。

各要素のおよその意味は以下の表の通りです。

＜セクション名と意味＞

番号	セクション名	意味	効果	割付先
1	.text	プログラムの命令コード	◎	OCPMEM
2	.bss	大域変数、static な局所変数	◎	OCDMEM
3	.cio			
4	.data	ASM 用データ領域		
5	.sysmem	ヒープ領域	◎	
6	.stack	システムスタック	◎	
7	.cinit	C 言語用初期化データ		
8	.const	大域定数、static な局所定数データ		
9	.switch	分岐テーブル	◎	
１０	.far	Far 参照用アドレステーブル	◎	

実行時間に大きく影響を及ぼすと思われるものは表の効果の覧に◎を付けてあります。

7.4　メモリー不足エラー例

メモリーが不足すると、RTW によるコード生成の最後の段階で、MATALB　Command Window にエラーメッセージが表示されます。

```
>> cannot allocate .text in OCPMEM (page 0)
>> cannot allocate .bss in OCDMEM (page 0)
>> cannot allocate .cinit in OCDMEM (page 0)
>> cannot allocate .sysmem in OCDMEM (page 0)
>> cannot allocate .far in OCDMEM (page 0)
>> errors in input - untitled.out not built
NMAKE : fatal error U1077: 'e:¥ti¥C6000¥CGTOOLS¥bin¥lnk6x.exe' : ﾘﾀｰﾝ ｺｰﾄﾞ
'0x1'
Stop.
```

>>cannot allcate の後ろに割り付けできなかったセクション名(.text 等)とその割付先（OCDMEM 等）が表示されています。

エラーが出た場合の対策は、表示されたセクション、又は、同じ割付先（OCPMEM／OCDMEM）の別のセクションを外部メモリーに割り当てます。

ＲＴモデル自動生成

第７版　2003.06.11

中部電機株式会社 ＣＳ事業部
TEL　0532-61-9566
FAX　0532-63-1081

ＭＡＴＬＡＢ 5.ｘ用
Ｉ／Ｏデバイスライブラリ

＜＜　目次　＞＞

1.　ハードウェアのセットアップ 5

1.1　Ｉ／Ｏボードアドレスの設定 5

2.　ＭＡＴＬＡＢ　５．ｘ版用ライブラリ 6

2.1　ブロックライブラリの概要 6

2.2　ブロックライブラリの開き方 6

2.3　ＡＤＳＰ６７４－００ デバイスドライバーブロック 7

2.3.1　概要 7

2.3.2　ホストＤＰ入力デバイスドライバーブロック 9

2.3.3　ホストＤＰ出力デバイスドライバーブロック 10

2.3.4　ホストＤＰ入出力デバイスの概念 11

2.3.5　ＳＩＯ入力デバイスドライバーブロック 12

2.3.6　ＳＩＯ出力デバイスドライバーブロック 14

2.3.7　ＳＩＯ初期化デバイスドライバーブロック 15

2.3.8　サーキュラーバッファ・デバイスドライバブロック 16

2.4　ＡＤＳＰ６７４－００ Ｏｎ Ｂｏａｒｄ ＰＩＯ ウィンドウ 17

2.4.1　ADSP674-00 ビット単位入力デバイスドライバブロック 19

2.4.2　ADSP674-00 ビット単位出力デバイスドライバブロック 20

2.4.3　ADSP674-00 終了時出力指定ビット単位出力デバイスドライバブロック 21

2.4.4　ADSP674-00 パラレル入力デバイスドライバブロック 22

2.4.5　ADSP674-00 パラレル出力デバイスドライバブロック 23

2.4.6　ADSP674-00 終了時出力指定パラレル出力デバイスドライバブロック 24

2.4.7　ADSP674-00 ポート入力デバイスドライバブロック 25

2.4.8　ADSP674-00 ポート出力デバイスドライバブロック 26

2.4.9　ADSP674-00 終了時出力指定 ポート出力デバイスドライバブロック 27

2.5　ＡＤＳＰ６７４－００ Ｏｎ Ｂｏａｒｄ ＰＩＯ AD&DA ウィンドウ 28

2.5.1　ADSP674-00 Ａ／Ｄ入力デバイスドライバブロック 29

2.5.2　ADSP674-00 Ｄ／Ａ出力デバイスドライバブロック 30

2.5.3　ADSP674-00 終了時出力指定Ｄ／Ａ出力デバイスドライバブロック 31

2.6　ＡＤＳＰ３２Ｘ－０３／５３デバイスドライバーブロック 32

2.6.1　概要 32

2.6.2　チャンネル番号とボードとの対応 34

2.6.3　１チャンネルＡ／Ｄ入力デバイスドライバブロック 35

2.6.4　多チャンネルＡ／Ｄ入力デバイスドライバブロック 36

2.6.5　１チャンネルD／A出力デバイスドライバブロック 37
2.6.6　多チャンネルD／A出力デバイスドライバブロック 38
2.6.7　終了時出力指定１チャンネルD／A出力デバイスドライバブロック 39
2.6.8　終了時出力指定 多チャンネルD／A出力デバイスドライバブロック 40
2.7　ＡＤＳＰ３２Ｘ－０６／５６デバイスドライバーブロック 41
2.7.1　概要 41
2.7.2　ポート／ビット番号とボードとの対応 43
2.7.3　ビット単位入力デバイスドライバブロック 45
2.7.4　ポート単位入力デバイスドライバブロック 46
2.7.5　ビット単位出力デバイスドライバブロック 47
2.7.6　ポート単位出力デバイスドライバブロック 48
2.7.7　複数ビット入力デバイスドライバブロック 49
2.7.8　複数ビット出力デバイスドライバブロック 50
2.7.9　終了時出力指定 ビット単位出力デバイスドライバブロック 51
2.7.10　終了時出力指定 ポート単位出力デバイスドライバブロック 52
2.8　ＡＤＳＰ３２Ｘ－１１／６１デバイスドライバーブロック 53
2.8.1　概要 53
2.8.2　ビット番号、ポート／チャンネル番号とボードとの対応 55
2.8.3　ビット単位、絶縁入力デバイスドライバブロック 57
2.8.4　ビット単位、絶縁出力デバイスドライバブロック 58
2.8.5　プリセット付きカウンタデバイスドライバブロック 59
2.8.6　カウンタプリセットデバイスドライバブロック 60
2.8.7　カウンタリセットデバイスドライバブロック 61
2.8.8　終了時出力指定 ビット単位、絶縁出力デバイスドライバブロック 62
2.8.9　複数ビット同時、絶縁入力デバイスドライバブロック 63
2.8.10　複数ビット同時、絶縁出力デバイスドライバブロック 64
2.9　ＡＤＳＰ３２Ｘ－１３／６３デバイスドライバーブロック 65
2.9.1　概要 65
2.9.2　チャンネル番号とボードとの対応 67
2.9.3　ＰＧＡバイパス 68
2.9.4　１チャンネルA／D入力デバイスドライバブロック 69
2.9.5　多チャンネルA／D入力デバイスドライバブロック 70
2.9.6　１チャンネルA／D入力（MP付き）デバイスドライバブロック 71
2.9.7　多チャンネルA／D入力（MP付き）デバイスドライバブロック 72
2.9.8　１チャンネルD／A出力デバイスドライバブロック 73
2.9.9　多チャンネルD／A出力デバイスドライバブロック 74

2.9.10　終了時出力指定 １チャンネルD／A出力デバイスドライバブロック 75
2.9.11　終了時出力指定 多チャンネルD／A出力デバイスドライバブロック 76
2.10　ＡＤＳＰ３２４－１４１デバイスドライバーブロック 77
2.10.1　概要 77
2.10.2　チャンネル番号とボードとの対応 79
2.10.3　１チャンネルA／D入力デバイスドライバブロック 80
2.10.4　多チャンネルA／D入力デバイスドライバブロック 81
2.10.5　１チャンネルD／A出力デバイスドライバブロック 82
2.10.6　多チャンネルD／A出力デバイスドライバブロック 83
2.10.7　終了時出力指定１チャンネルD／A出力デバイスドライバブロック 84
2.10.8　終了時出力指定 多チャンネルD／A出力デバイスドライバブロック 85
2.11　ＡＤＳＰ３２４－１４３デバイスドライバーブロック 86
2.11.1　概要 86
2.11.2　コネクタとボードとのメモリマップ 87
2.11.3　ＣＮ１２　Ｉｎｐｕｔデバイスドライバブロック 88
2.11.4　ＣＮ１３　Ｉｎｐｕｔデバイスドライバブロック 89
2.11.5　ＤＳＰ間ＤＰ通信アービトレーション 90
2.11.6　ＤＳＰの前段と後段の意味 91
2.12　ＡＤＳＰ３２４－１４５デバイスドライバーブロック 92
2.12.1　概要 92
2.12.2　コネクタとボードとのメモリマップ 93
2.12.3　WatchDogデバイスドライバブロック 95
2.12.4　Pulse outputデバイスドライバブロック 96
2.12.5　Counter lnputデバイスドライバブロック 97
2.12.6　Counter lnput with Change reset　flagデバイスドライバブロック 98
2.12.7　Counter presetデバイスドライバブロック 99
2.12.8　Capture(S) デバイスドライバブロック 100
2.12.9　Capture(3)デバイスドライバブロック 101
2.12.10　Pwm Output L〔S〕デバイスドライバブロック 102
2.12.11　Pwm Output L (3)デバイスドライバブロック 103
2.12.12　Pwm Output L (S) with Freqデバイスドライバブロック 104
2.12.13　Pwm Output L (3) with Freqデバイスドライバブロック 105
2.12.14　PIO bit In〔32〕デバイスドライバブロック 106
2.12.15　PIO bit In〔16〕デバイスドライバブロック 107
2.12.16　PIO paralell in〔32〕デバイスドライバブロック 108
2.12.17　PIO paralell in〔16〕デバイスドライバブロック 109

2.12.18 PIO bit out〔32〕デバイスドライバブロック 110
2.12.19 PIO bit out〔16〕デバイスドライバブロック 111
2.12.20 PIO bit out with Terminate〔32〕デバイスドライバブロック 112
2.12.21 PIO bit out with Terminate〔16〕デバイスドライバブロック 113
2.12.22 PIO paralell out〔32〕デバイスドライバブロック 114
2.12.23 PIO paralell out〔16〕デバイスドライバブロック 115
2.12.24 PIO parallel out with Terminate〔32〕デバイスドライバブロック 116
2.12.25 PIO paralell out with Terminate〔16〕デバイスドライバブロック 117
2.12.26 From GOTデバイスドライバブロック 118
2.12.27 From GOT[8]デバイスドライバブロック 119
2.12.28 From GOT[16]デバイスドライバブロック 120
2.12.29 From GOT32 デバイスドライバブロック 121
2.12.30 From GOT32 [8]デバイスドライバブロック 122
2.12.31 From GOT32 [16]デバイスドライバブロック 123
2.12.32 To GOTデバイスドライバブロック 124
2.12.33 To GOT[8]デバイスドライバブロック 125
2.12.34 To GOT[16]デバイスドライバブロック 126
2.12.35 To GOT32 デバイスドライバブロック 127
2.12.36 To GOT32 [8]デバイスドライバブロック 128
2.12.37 To GOT32 [16]デバイスドライバブロック 129
2.13 ＴＭ３２ＤＡデバイスドライバーブロック 130
2.13.1 概要 130
2.13.2 １チャンネルＤ／Ａ出力デバイスドライバブロック 132
2.13.3 多チャンネルＤ／Ａ出力デバイスドライバブロック 133
2.14 Data I/O By Excelブロック 134
2.14.1 概要 134
2.14.2 From Excelブロック 135
2.14.3 To Excelブロック 137

3. 他のＤＳＰとの互換性 138
3.1 他のDSPとの互換性 138

1. ハードウェアのセットアップ

1.1 Ｉ／Ｏボードアドレスの設定

ＤＳＰボード（ＡＤＳＰ４０４－００）はＩ／Ｏボードの種類を問わず最大４台のＩ／Ｏボードを接続できるように設計されています。本製品はそれに対応し、合計４台までの各種Ｉ／Ｏボードを接続した場合を想定しています。各Ｉ／Ｏボードはアドレスで識別しますので、Ｉ／Ｏアドレスの設定を適切に行なわなければいけません。

まず、各Ｉ／Ｏボードに個々のボードを識別する為の番号を、その種類毎に１台目から割り付ける事とします。例えば４種類のボードを各１台ずつ接続したとすると、何れも１台目となります。又、同じ種類のボードを複数接続する場合は、ボードの種類別に番号を飛ばさないように１台目から順に割り付けます。こうして割り付けた各ボードに対して 表 1.1.1 の様にボードアドレスを設定して下さい。ボードアドレスの具体的設定方法は各Ｉ／Ｏボードのハードウェア・ユーザーズ・マニュアルを参照して下さい。

表 1.1.1　ボードアドレス設定表

	Ｉ／Ｏボードの種類			
	ADSP324-03	ADSP324-06	ADSP324-11	ADPS324-13
１台目	900000h	900080h	900100h	900180h
２台目	900010h	900090h	900110h	900190h
３台目	900020h	9000A0h	900120h	9001A0h
４台目	900030h	9000B0h	900130h	9001B0h

	TM32DA	ADSP324-141	ADSP324-143	ADPS324-145
１台目	900200h	900280h	908000h	901000h
２台目	900220h	9002A0h	90C000h	901200h
３台目	900240h	9002C0h	910000h	901400h
４台目	900260h	9002E0h	914000h	901600h

2. ＭＡＴＬＡＢ ５．ｘ版用ライブラリ

2.1 ブロックライブラリの概要

本製品のブロックライブラリは、ＳＩＭＵＬＩＮＫの標準ブロックや、オプションのブロックセットと同様に、ブロック線図に貼り付けて使用します。こうしたブロック線図をReal-Time Workshop でコード化し、ＤＳＰ上で実行する際にこれらブロックにより、外部Ｉ／Ｏとデータ交換が出来ます。

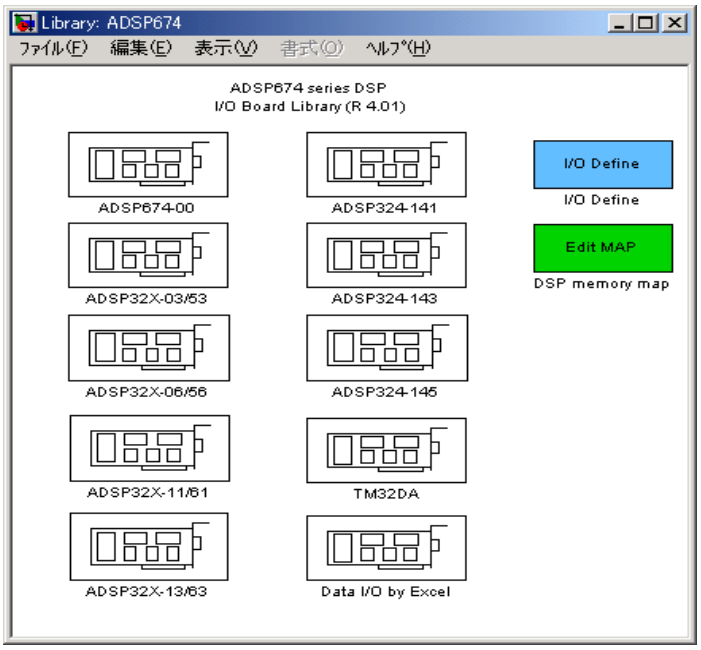
2.2 ブロックライブラリの開き方

ブロックライブラリは、ＭＡＴＬＡＢコマンドウィンドウに次のコマンドを入力します。

>> adsp674

すると、図 2.2-1　ＡＤＳＰ６７４ウインドウに示すブロックライブラリが開きます。

図 2.2-1　ＡＤＳＰ６７４ウインドウ



ＡＤＳＰ６７４ウィンドウには、現在標準で供給されているＤＳＰ用のＩ／Ｏボード及びＤＳＰボード本体のＩ／Ｏに関するデバイスドライバーをボード毎に分類して収納してあります。

各Ｉ／Ｏボードの型番を記したアイコン（ADSP32X-03/53 等）が登録されています。これらアイコンをダブルクリックすることにより、Ｉ／Ｏボードに対応したサブブロックのウィンドウを開きます。

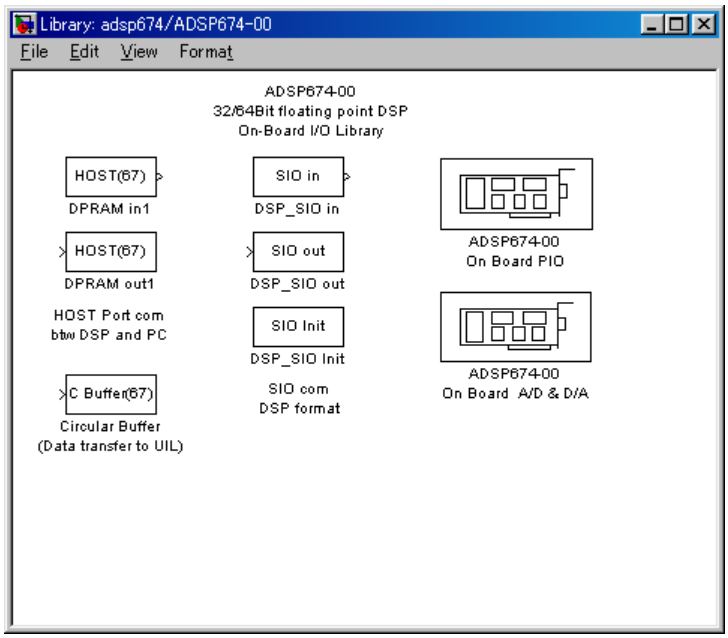
また、各モデル毎にＩ／Ｏ Ｄｅｆｉｎｅブロックが必要です。このブロックを１モデルに１ブロック使用してください。内容は、実装するＩ／Ｏボードの枚数を設定するブロックです。このブロックの内容によりＩ／Ｏをアクセス制御します。

2.3 ＡＤＳＰ６７４－００ デバイスドライバーブロック

2.3.1　概要

図 2.3-1に、ＡＤＳＰ６７４－００（ＤＳＰボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.3-1　ADSP674-00 ウィンドウ



この中には ＤＳＰボード上のＩ／Ｏ（ＤＰＲＡＭ等含む）とのインターフェースを取るためのデバイスドライバーブロックが登録されています。

HOST(67)	ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストとの間で非同期にデータ交換をする為のブロックです。
C Buffer(67)	ＤＳＰのサーキュラーバッファ機構を使った高速データバッファリングによるホストへのデータ送信を行うブロックです。高速に処理させたいブロック線図で使用する事により、データをバッファリングする為の負担を低減させることが出来ます。バッファリングしたデータは、ＵＩＬを用いたユーザアプリケーションへデータで取り出す事が出来ます。
SIO in/out/init	ＤＳＰのＳＩＯを使ったシリアル送受信を行うブロックです。ＤＳＰボード間での通信に使用します。

ADSP674-00 on Board PIO

ＤＳＰボード上のＰＩＯをアクセスするブロックを収納してある ADSP674-00 On Bard PIO ウィンドウを開きます。

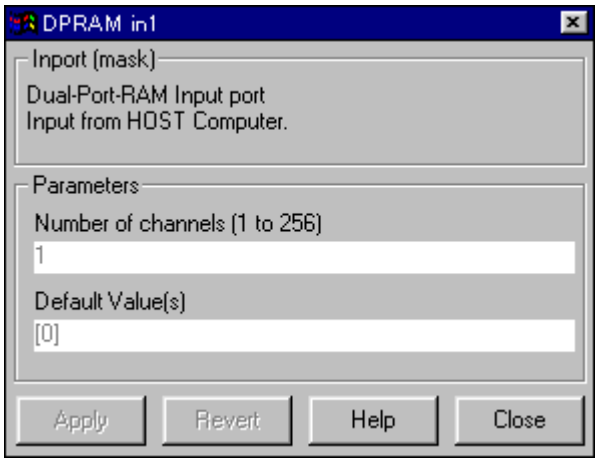
ADSP674-00 on Board AD & DA

ＤＳＰボード上のアナログＩＯ（ＡＤ及びＤＡ）をアクセスするブロックを収納してある ADSP674-00 On Bard AD & DA ウィンドウを開きます。

2.3.2　ホストＤＰ入力デバイスドライバーブロック

ホストＤＰ入力デバイスドライバーブロックは、ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホスト－ＤＳＰ間で非同期にデータ交換を行う為のブロックです。主に、ＵＩＬを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、2.3.4 ホストＤＰ入出力デバイスの概念 を参照してください。

図 2.3-2 HOST DPRAM in



- Number of Channels (1 to 256)

  ホストから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Value(s)

  ホストが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

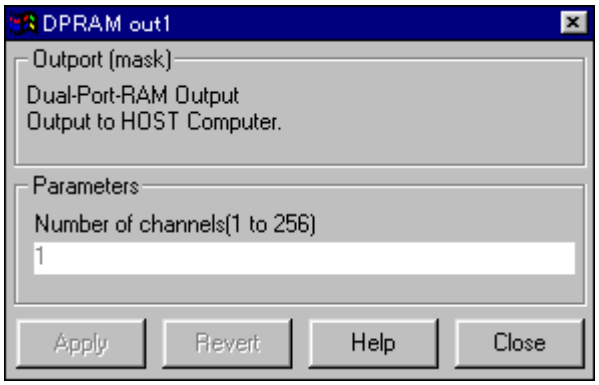
このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、３８００Ｈ番地から連続する２５６ワードを使っています。３８００Ｈ番地がベクトルの最初の要素です。データ形式は、ＤＳＰの実数フォーマットですので、ＵＩＬ以外のアプリケーションで通信する場合はホスト側でＩＥＥＥとの間のデータ変換が必要です。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとＤＳＰのアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

このブロックは１つのブロック線図に１個しか配置できません。

2.3.3　ホストＤＰ出力デバイスドライバーブロック

ホストＤＰ出力デバイスドライバーブロックは、ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストーＤＳＰ間で非同期にデータ交換を行う為のブロックです。主に、ＵＩＬを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、2.3.4 ホストＤＰ入出力デバイスの概念 を参照してください。

図 2.3-3　HOST DPRAM out



- Number of Channels (1 to 256)

  ホストへ送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルにまとめてからこのブロックに入力します。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、３Ｃ００Ｈ番地から連続する２５６ワードを使っています。３Ｃ００Ｈ番地がベクトルの最初の要素です。データ形式は、ＤＳＰの実数フォーマットですので、ＵＩＬ以外のアプリケーションで通信する場合はホスト側でＩＥＥＥとの間のデータ変換が必要です。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとＤＳＰのアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

このブロックは１つのブロック線図に１個しか配置できません。

2.3.4　ホストＤＰ入出力デバイスの概念

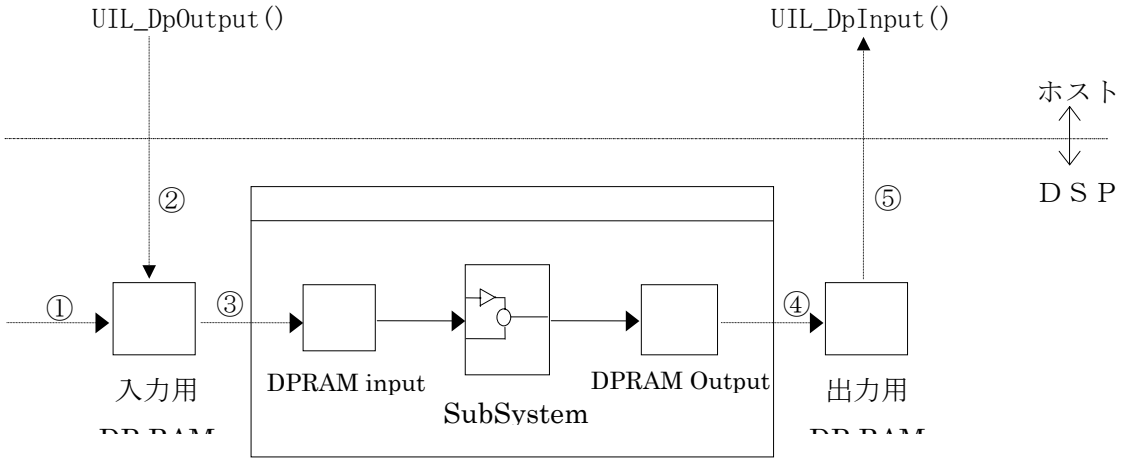
ホストＤＰＲＡＭインターフェースは、ＤＳＰボードに実装されているデュアルポートＲＡＭ（ＤＰＲＡＭ*１参考）を使い、ホストのユーザインターフェースライブラリとＤＳＰ内のＲＴＷによるリアルタイムモデルとの間で、非同期なデータ交換を可能にするものです。この機能を用いて相互にデータ転送が可能です。

この機能の実装は次の２つの作業で可能となります。

１、ホスト側のユーザプログラムにＵＩＬライブラリをリンク。

２、ＤＳＰ側の SIMUINK ブロック線図にＤＰライブラリを配置。

次の図はＤＰ－ＲＡＭインターフェースのデータ交換の流れを記した図です。



①リアルタイムモデルの初期化時にセットされます。

②ホストプログラムから関数呼び出し時にセットされます。

③ＲＴモデルのステップサイズ毎に読み出しされます。

④ＲＴモデルのステップサイズ毎に書き込まれます。

⑤ホストプログラムから関数呼び出し時点のデータが読み出されます。

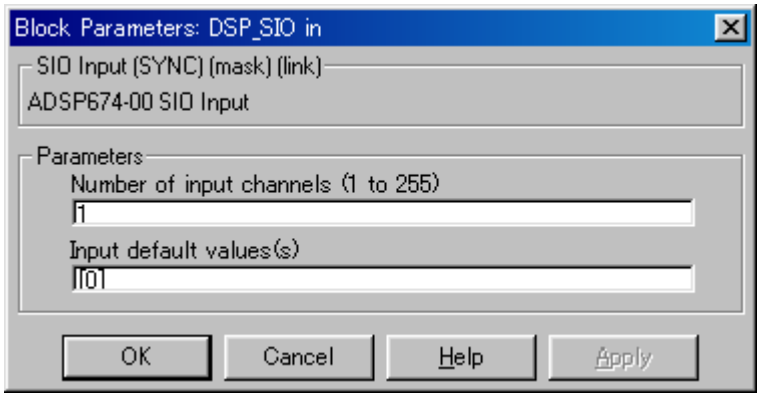
＊１参考：デュアルポートＲＡＭ（ＤＰＲＡＭ）はＤＳＰボードに標準実装されている特殊なメモリーで、ＤＳＰとホストコンピュータの間で非同期、かつ、双方向に通信が可能です。

2.3.5　ＳＩＯ入力デバイスドライバーブロック

ＳＩＯ入力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使い、他のＤＳＰボードと通信します。

このブロックを使用するには、ＳＩＯ初期化ブロックを同じブロック線図に配置する必要があります。

図 2.3-4　SIO in



- Number of input Channels (1 to 255)

  受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Value(s)

  相手が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックから入力されるデータは、送信元ＤＳＰが送信した最も新しいデータです。複数チャンネルのデータを受け取る場合、チャンネル間のデータの同時性は保証されません。

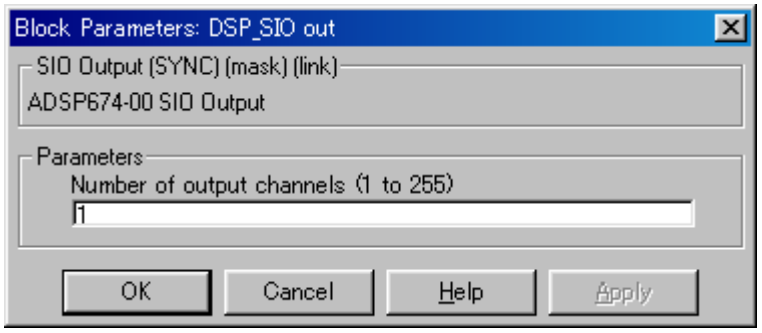
受け取る側と相手側から送信されて来るデータのチャンネル数の整合は、確認されません。もし、相手ＤＳＰが送信したデータの点数が Number of input Channels の設定を超える場合、超えた分のデータは破棄されます。逆に、Number of input Channels で指定された点数のデータが送られない場合は、不足するチャンネルからは、Default Value(s) で設定したデフォルト値を出力します。例えば受信側が５チャンネルで送信側が３チャンネルの場合、先頭側の３チャンネルには相手から送られたデータが、そして残りは Default Value(s) で設定した値が入力されます。

データはベクトルで出力されますので、個々のチャンネルのスカラーデータが必要な場合は、Demux 等で分解して利用します。

2.3.6　ＳＩＯ出力デバイスドライバーブロック

ＳＩＯ出力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使い、他のＤＳＰ（ＡＤＳＰ６７４－００）に対してデータを送信します。
このブロックを使用するには、ＳＩＯ初期化ブロックを同じブロック線図に配置する必要があります。

図 2.3-5　SIO out



- Number of output Channels (1 to 255)

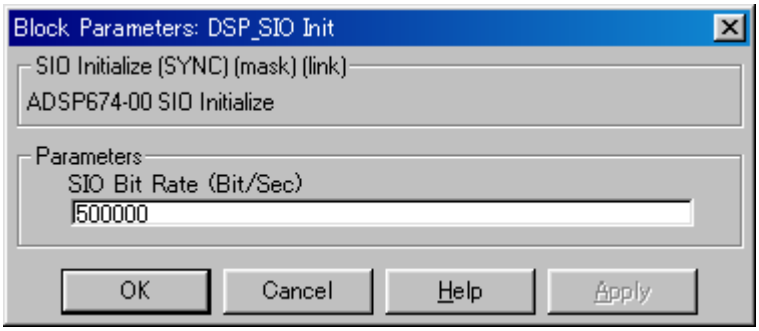
  送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとをこのブロックに入力します。

このブロックに出力されるデータは、ＳＩＯポートに自動的に送信されます。
このブロックへの出力データはベクトルで与えます。複数のスカラーデータを出力する場合は、Mux ブロック等でベクトルにまとめてから出力します。

2.3.7　ＳＩＯ初期化デバイスドライバーブロック

ＳＩＯ初期化デバイスドライバーブロックは、ＳＩＯ入力・出力デバイスドライバーブロックを使う場合に、ＳＩＯポートを初期化する為使います。

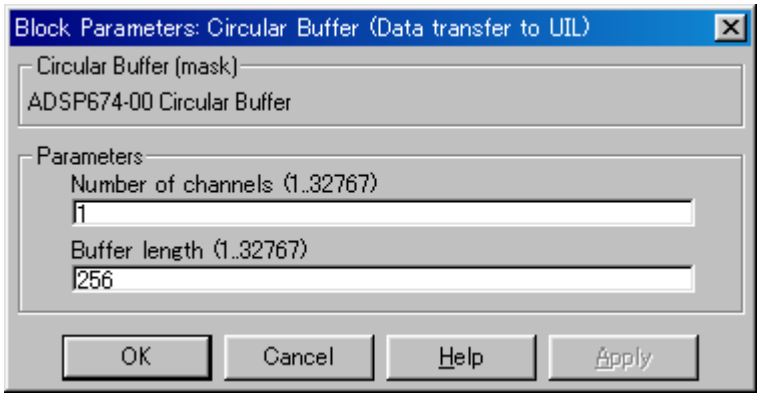
図 2.3-6　SIO Init



2.3.8　サーキュラーバッファ・デバイスドライバブロック

ＤＳＰのメモリー内に高速にデータ保管が行えるサーキューラーバッファを構築し、ここにデータを貯える為のブロックです。貯えたデータはＵＩＬ（オプション）にて取り出します。

図 2.3-7　C Buffer(67)



- Number of Channels (1 .. 32767)

  このブロックへの入力データの点数を指定します。入力データはベクトルで与えますので、このベクトルの幅と同じです。

- Buffer length (1 .. 32767)

  各データの保存する長さを指定します。

データバッファはＤＳＰ内のメモリーに確保されます。その概ねの長さは、

バッファ長さ　＝　（データ点数＋１）×（データの長さ＋α）

αはバッファの長さの最小単位が２のＮ乗である事から、その最小単位に切り上げる為の係数です。この領域は、実行時にヒープ領域から確保されますので、充分な長さのヒープ領域が必要です。

バッファが確保出来たかどうかは、ＵＩＬで確認できます。バッファが確保出来なかった場合は、データの保管は行われません。その場合、データ保管する長さを減らす等の再試行は、ブロック線図の変更とＲＴＷの再実行により行ってください。

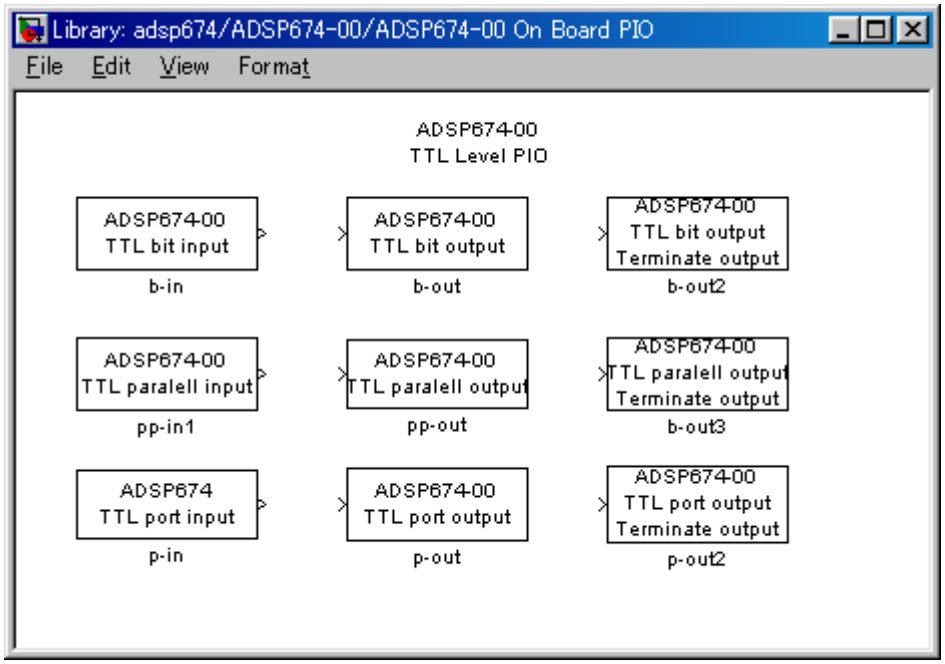
バッファは、サーキュラーバッファと呼ばれる、輪状につながったエンドレスのバッファを構築しています。バッファ内のデータが満杯になった段階で、新たなデータは最古のデータに上書きされ、最新の Buffer length 点のデータが、バッファに残ります。

データはＵＩＬにて取り出します。現在何点のデータが格納されているか等の情報は、ＵＩＬで取り出すことができます。

2.4 ＡＤＳＰ６７４－００ Ｏｎ Ｂｏａｒｄ ＰＩＯ ウィンドウ

ADSP674-00 ウィンドウの ADSP674-00 On Board PIO をダブルクリックして開くウィンドウです。ここには DSP ボード上に実装されている PIO ポートをアクセスする IO デバイスドライバーが収納されています。

図 2.4-1 ADSP674-00 On Board PIO



- ADSP674-00 TTL bit input

  １ビット単位で入力を行うブロックです。

- ADSP674-00 TTL bit output

  １ビット単位で出力を行うブロックです。

- ADSP674-00 TTL bit output Terminate output

  １ビット単位で出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

- ADSP674-00 TTL parallel input

  全ビットをパラレルに入力するブロックです。

- ADSP674-00 TTL parallel output

  全ビットをパラレルに出力するブロックです。

- ADSP674-00 TTL parallel output Terminate output
  全ビットをパラレルに出力するブロックです。モデルの動作を終了する際に予め指定した状態を出力します。
- ADSP674-00 TTL port input
  ポート単位で入力を行うブロックです。
- ADSP674-00 TTL port output
  ポート単位で出力を行うブロックです。
- ADSP674-00 TTL port output Terminate output
  ポート単位で入力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

表 2.4.1　ビット番号とＡＤＳＰ６７４－００ボードPI0との対応

入出力信号名	ビット番号 注１	パラレル 注２	ポート単位 注３
EXTIN7/ EXTOUT7	7	最後	2^7
EXTIN6/ EXTOUT6	6	最後の１つ前	2^6
EXTIN5/ EXTOUT5	5	・	2^5
EXTIN4/ EXTOUT4	4	・	2^4
EXTIN3/ EXTOUT3	3	・	2^3
EXTIN2/ EXTOUT2	2	・	2^2
EXTIN1/ EXTOUT1	1	先頭の次	2^1
EXTIN0/ EXTOUT0	0	先頭	2^0

注１：ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

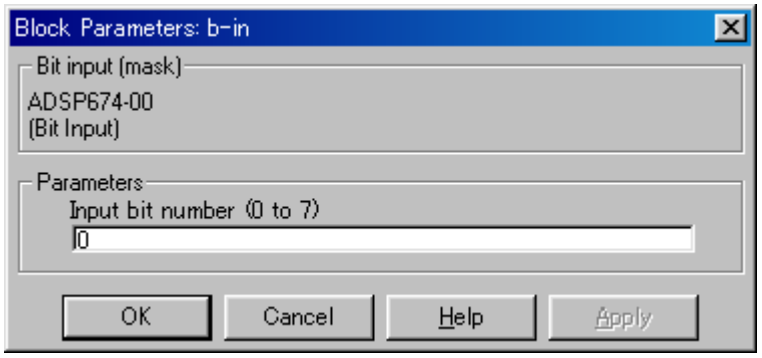
注２：パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注３：ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

2.4.1　ADSP674-00 ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時０、HIGH の時１です。入力信号が接続されていないビットは１になります。

図 2.4-2　ADSP674-00 TTL bit input



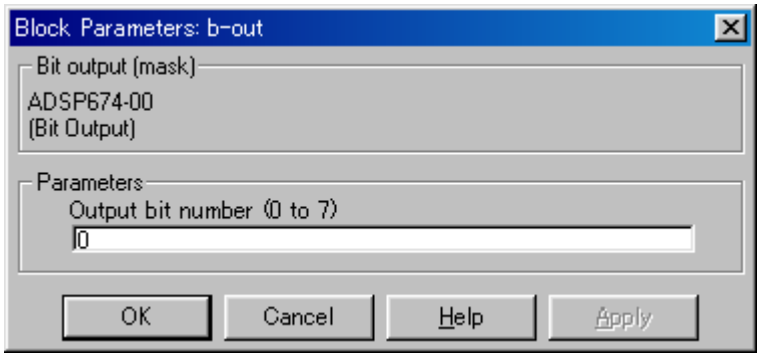
- Input bit number (0 to 7)

  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1 (18ページ) を参照してください。

2.4.2　ADSP674-00 ビット単位出力デバイスドライバブロック

ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが 0 の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 2.4-3　ADSP674-00 TTL bit output



- Output bit number (0 to 7)

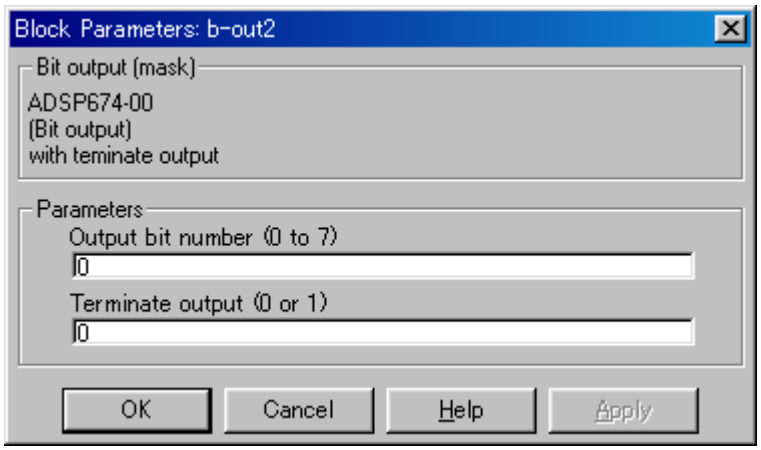
  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1 (18ページ) を参照してください。

2.4.3　ADSP674-00 終了時出力指定ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 2.4-4　ADSP674-00 TTL bit output with Terminate output



- Output bit number (0 to 7)

  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

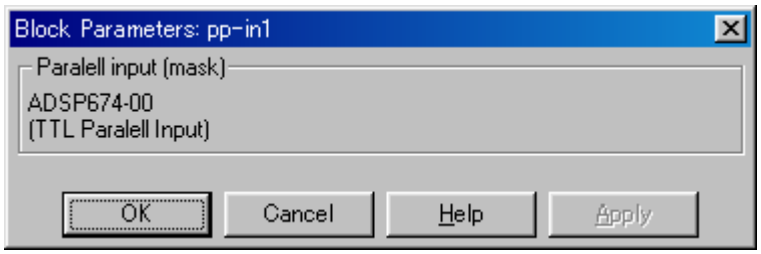
- Terminate output　(0 or 1)

  実時間モデルが停止する際に出力して欲しい状態を設定します。０でＬＯＷ，１でＨＩＧＨが出力されます。

2.4.4　ADSP674-00 パラレル入力デバイスドライバブロック

パラレル入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの数と同じ点数のベクトルとして入力します。ボードに入力される信号レベルが LOW の時、これに該当する信号は 0 となります。入力が接続されていない信号の値は不定です。

図 2.4-5　ADSP674-00 TTL paralell input



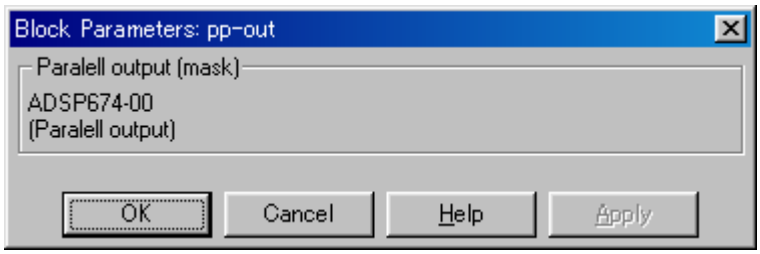
このブロックの初期化ダイアログには設定すべき項目はありません。
入力信号のビット位置とハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.5　ADSP674-00 パラレル出力デバイスドライバブロック

パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が 0 の場合は LOW を、それ以外の場合は HIGH を出力します。

図 2.4-6　ADSP674-00 TTL paralell output



このブロックの初期化ダイアログには設定すべき項目はありません。
出力信号のビットの位置とハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

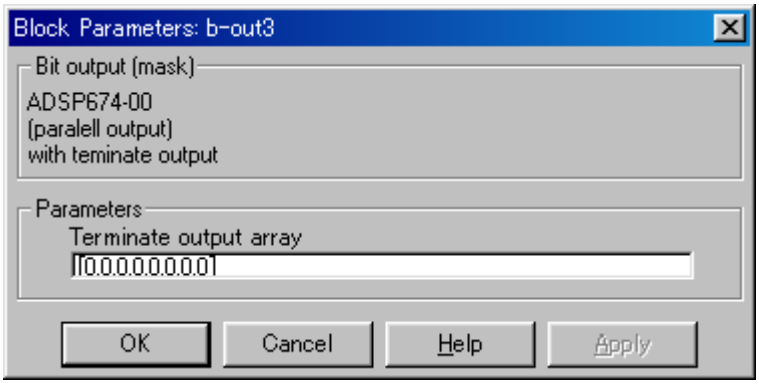
注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.6　ADSP674-00 終了時出力指定パラレル出力デバイスドライバブロック

終了時出力指定パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が 0 の場合は LOW を、それ以外の場合は HIGH を出力します。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.4-7　ADSP674-00 TTL paralell output



- Terminate output array

  実時間モデルが停止する際に出力して欲しい値をベクトルで指定します。

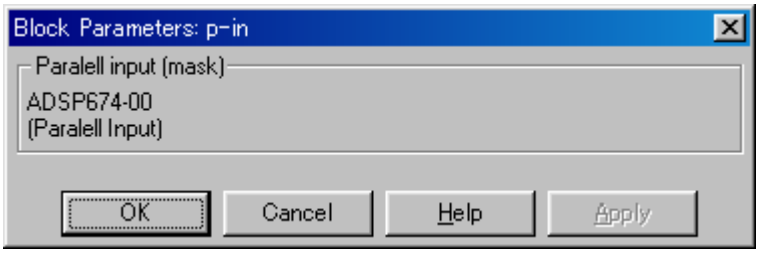
出力信号のビットの位置とハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.7　ADSP674-00 ポート入力デバイスドライバブロック

ポート入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの重みをかけて加算された値を入力します。即ち PIO 入力をバイナリ値として読み込みます。ボードに入力される信号レベルが LOW の時、これに該当するビットは 0 となります。入力が接続されていない信号の値は不定です。

図 2.4-8　ADSP674-00 TTL port input



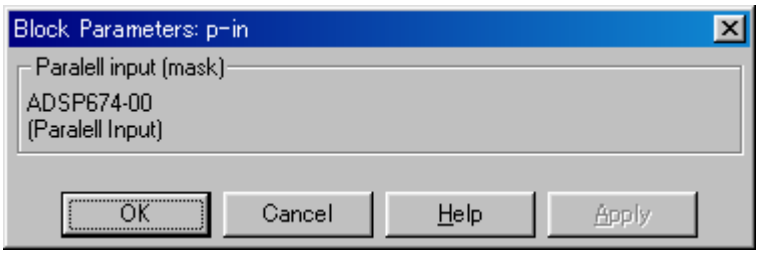
このブロックの初期化ダイアログには設定すべき項目はありません。
入力信号のビットの重みとハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

2.4.8　ADSP674-00 ポート出力デバイスドライバブロック

ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが 0 の場合、ボードに出力される信号レベルは LOW となります。

ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。

図 2.4-9　ADSP674-00 TTL port　output



このブロックの初期化ダイアログには設定すべき項目はありません。

出力信号のビットの重みとハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

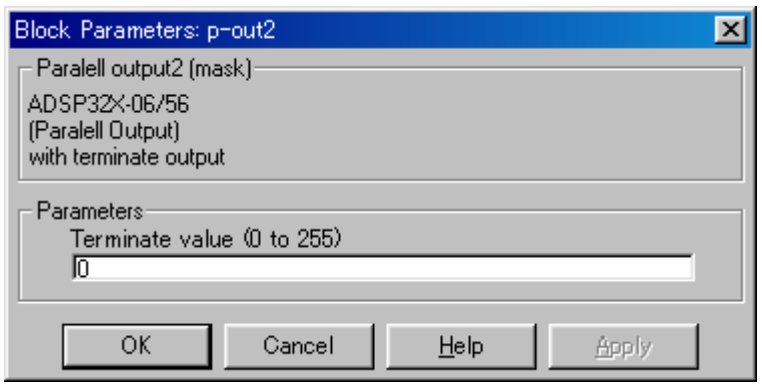
2.4.9　ADSP674-00 終了時出力指定 ポート出力デバイスドライバブロック

終了時出力指定ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが０の場合、ボードに出力される信号レベルは LOW となります。

ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.4-10　ADSP674-00 TTL port　output　terminate output



- Terminate value (0 to 255)

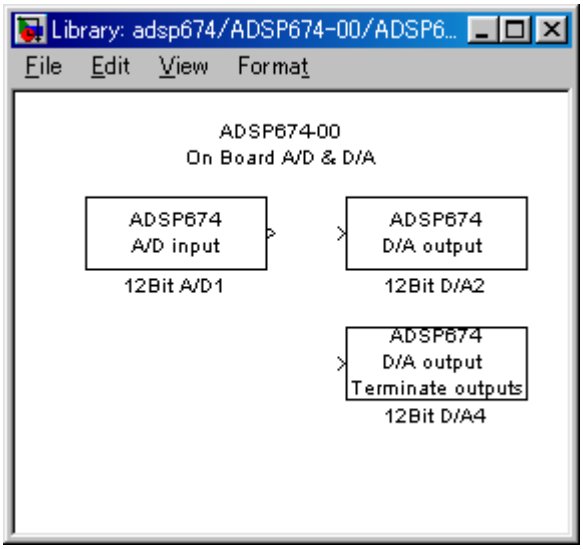
  実時間モデルが停止する際に出力して欲しい値を１０進数で設定します。

出力信号のビットの重みとハードウェアとの対応は、表 2.4.1（18ページ）を参照してください。

2.5 ＡＤＳＰ６７４－００ Ｏｎ Ｂｏａｒｄ ＰＩＯ AD&DA ウィンドウ

ADSP674-00 ウィンドウの ADSP674-00 On Board AD & DA をダブルクリックして開くウィンドウです。ここには DSP ボード上に実装されている AD や DA ポートをアクセスする IO デバイスドライバーが収納されています。

図 2.5-1 ADSP674-00 On Board AD & DA



- ADSP674-00 AD input

  AD から入力を行うブロックです。

- ADSP674-00 DA output

  DA への出力を行うブロックです。

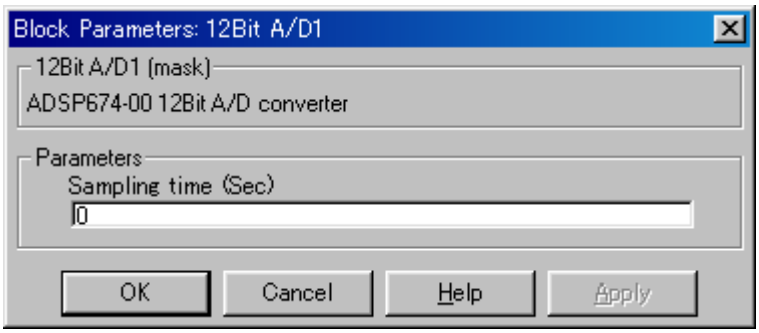
- ADSP674-00 DA output Terminate output

  DA への出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

2.5.1　ADSP674-00　A／D入力デバイスドライバブロック

A／D入力デバイスドライバブロックは、DSP ボード上ののA／Dから一括してデータを入力します。入力の対象となるA／Dチャンネルは、このブロックからの出力は、２の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.5-2　ADSP674-00 A/D input



- Sampling time (Sec)

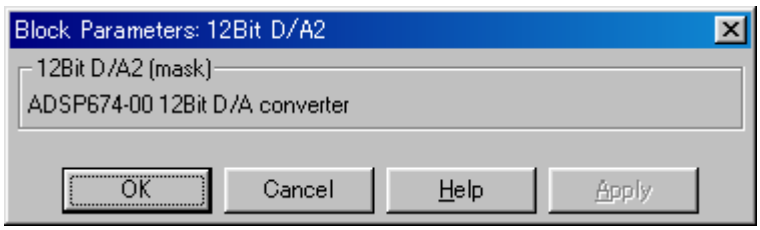
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA／D変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にA／D変換が行われます。

2.5.2　ADSP674-00 Ｄ／Ａ出力デバイスドライバブロック

Ｄ／Ａ出力デバイスドライバブロックは、幅２のベクトルデータを受取り、Ｄ／Ａへ一括して出力します。このブロックへの入力ベクトルの幅は必ず２でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のＤ／Ａへ出力されます。

このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.5-3　ADSP674-00 D/A output



このブロックの初期化ダイアログには設定すべき項目はありません。

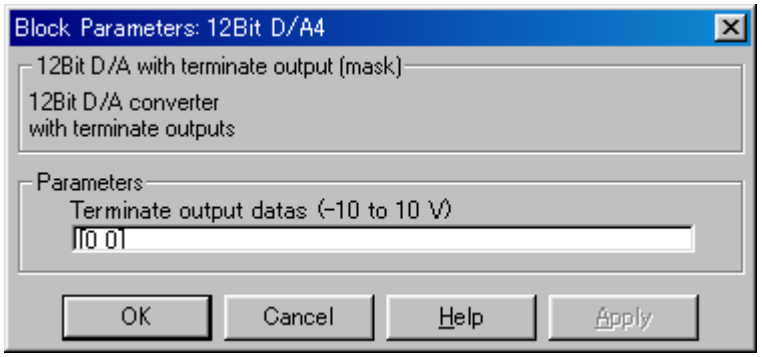
2.5.3　ADSP674-00 終了時出力指定Ｄ／Ａ出力デバイスドライバブロック

終了時出力指定Ｄ／Ａ出力デバイスドライバブロックは、幅２のベクトルデータを受取り、Ｄ／Ａへ一括して出力します。このブロックへの入力ベクトルの幅は必ず２でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のＤ／Ａへ出力されます。

このブロックはモデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.5-4　ADSP674-00 D/A output



- Terminate output datas　(-10 to 10V)

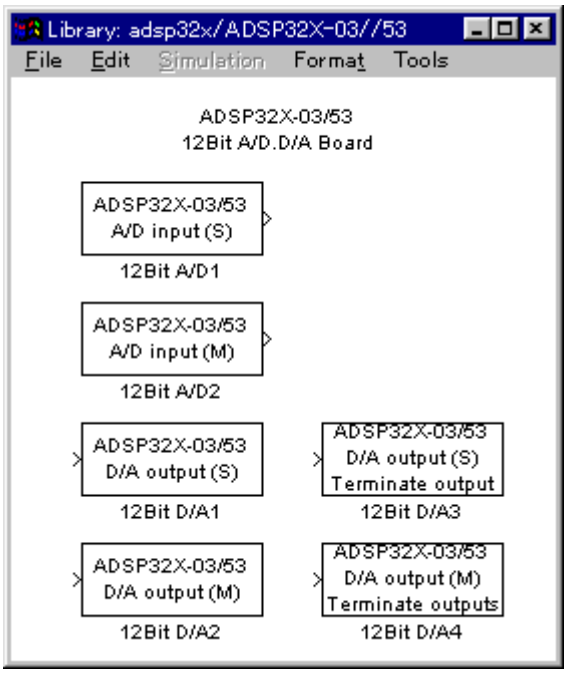
  リアルタイムモデルが終了する際にＤ／Ａから出力する値をベクトルで設定します。単位は[Volt]です。

2.6 ＡＤＳＰ３２Ｘ－０３／５３デバイスドライバーブロック

2.6.1 概要

図 2.6-1に、ＡＤＳＰ３２Ｘ－０３／５３（１２ビットＡ／Ｄ・Ｄ／Ａボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.6-1　ADSP32X-03/53 ドライバウィンドウ



この中にはＡＤＳＰ３２Ｘ－０３／５３とのインターフェースを取るための４つのデバイスドライバーブロックが登録されています。

- A/D input (S)　　１チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
- A/D input (M)　　多チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
- D/A input (S)　　１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M)　　多チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (S) Terminate output
  終了時の出力指定が行える、１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時の出力指定が行える、多チャンネルのＤ／Ａ出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、ＵＩＬ、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのＡ／Ｄ入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。Ｄ／Ａの場合も同様です。特に、Ａ／Ｄでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.6.2　チャンネル番号とボードとの対応

ＡＤＳＰ３２Ｘ－０３／５３は最大４台使用でき、１台のボードにＡ／Ｄ、Ｄ／Ａが各４チャンネル実装されています。ＡＤＳＰ３２Ｘ－０３／５３の各デバイスドライバブロックはこれらのチャンネルを０～１５の連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 2.6.1の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

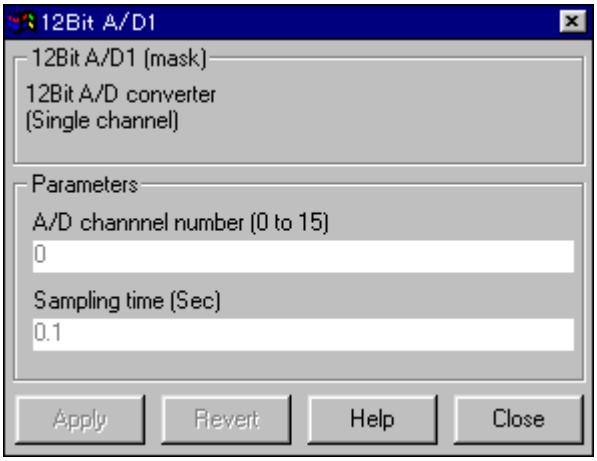
表 2.6.1　チャンネル番号とＡＤＳＰ３２Ｘ－０３／５３ボードとの対応

ボード	IN/OUT0	IN/OUT1	IN/OUT2	IN/OUT3
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

2.6.3　１チャンネルＡ／Ｄ入力デバイスドライバブロック

１チャンネルＡ／Ｄ入力デバイスドライバブロックは、指定されたチャンネル番号のＡ／Ｄからデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.6-2　ADSP32X-03/53 A/D input (S)



- A/D channel number (0 to 15)

  入力するＡ／Ｄのチャンネル番号を０～１５の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1（34ページ）を参照してください。

- Sampling time (Sec)

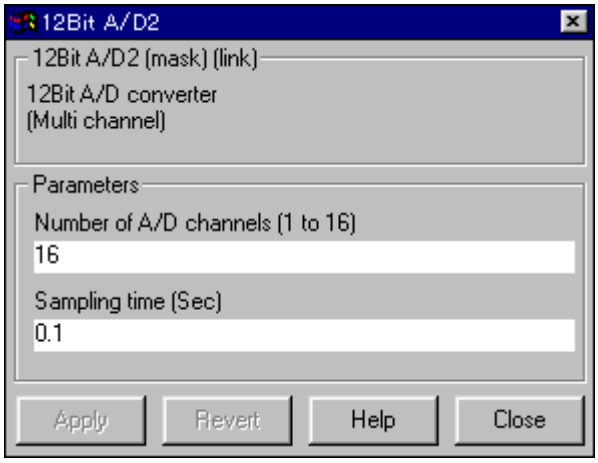
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

2.6.4　多チャンネルＡ／Ｄ入力デバイスドライバブロック

多チャンネルＡ／Ｄ入力デバイスドライバブロックは、複数チャンネルのＡ／Ｄから一括してデータを入力します。入力の対象となるＡ／Ｄチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。

このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.6-3　ADSP32X-03/53 A/D input (M)



- Number of A/D channels (1 to 16)

  入力するＡ／Ｄのチャンネル数を１～１６の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1（34ページ）を参照してください。

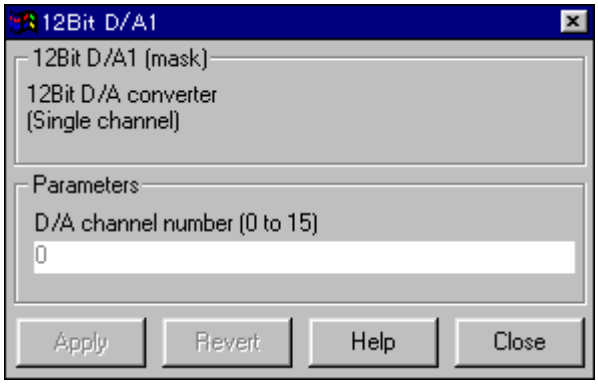
- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

2.6.5　１チャンネルＤ／Ａ出力デバイスドライバブロック

１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-4　ADSP32X-03/53 D/A output (S)



- D/A channel number (0 to 15)

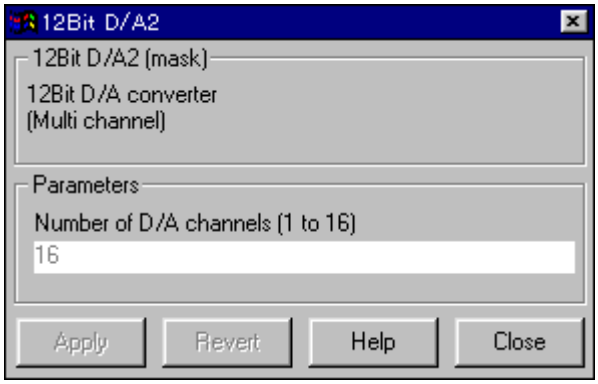
  出力するＤ／Ａのチャンネル番号を０～１５の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1（34ページ）を参照してください。

2.6.6　多チャンネルＤ／Ａ出力デバイスドライバブロック

多チャンネルＤ／Ａ出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのＤ／Ａへ一括して出力します。出力の対象となるＤ／Ａチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のＤ／Ａへ出力されます。

このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-5　ADSP32X-03/53 D/A output (M)



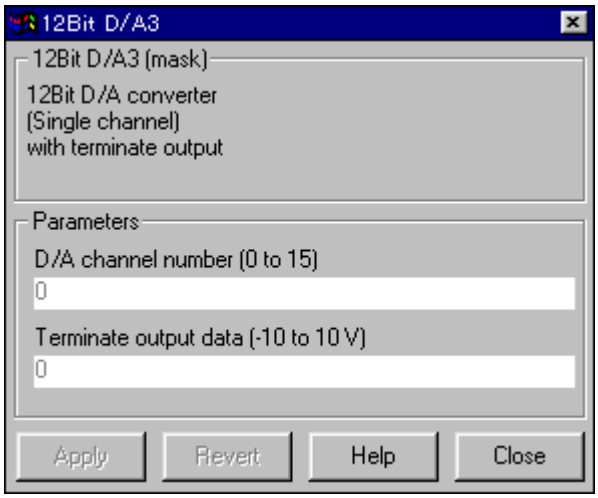
- Number of A/D channels (1 to 16)

  出力するＤ／Ａのチャンネル数を１～１６の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (34ページ) を参照してください。

2.6.7　終了時出力指定１チャンネルＤ／Ａ出力デバイスドライバブロック

終了時出力指定付き１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-6　ADSP32X-03/53 D/A output (S) Terminate output



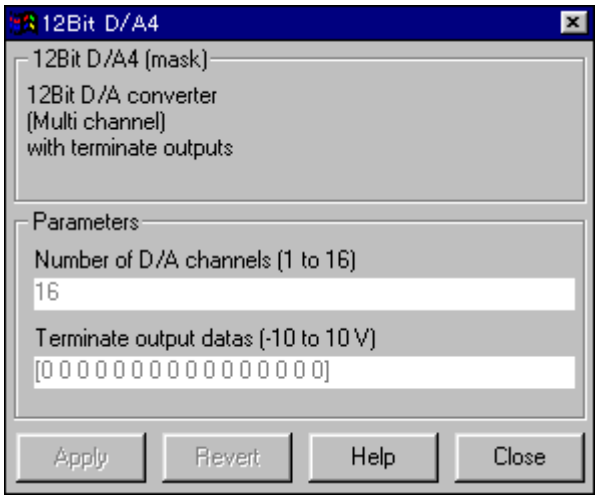
- D/A channel number (0 to 15)
  出力するＤ／Ａのチャンネル番号を０～１５の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (34ページ) を参照してください。
- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際にＤ／Ａから出力する値を設定します。単位は[Volt]です。

2.6.8　終了時出力指定 多チャンネルD／A出力デバイスドライバブロック

終了時出力指定付き多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD／Aへ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-7　ADSP32X-03/53 D/A output (M) Terminate output



- Number of D/A channels (1 to 16)

  出力するD／Aのチャンネル数を１～１６の整数で指定します。先頭チャンネル番号は0で固定です。チャンネル番号とハードウェアとの対応は、表 2.6.1 (34 ページ) を参照してください。

- Terminate output data (-10 to 10V)

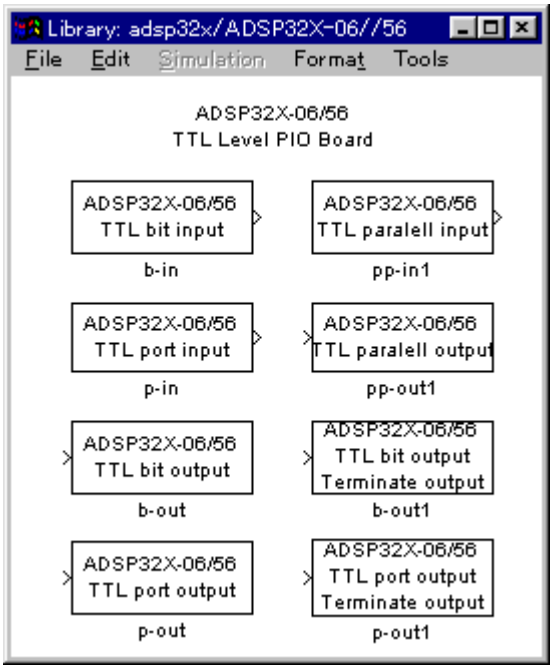
  リアルタイムモデルが終了する際にD／Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.7 ＡＤＳＰ３２Ｘ－０６／５６デバイスドライバーブロック

2.7.1 概要

図 2.7-1に、ＡＤＳＰ３２Ｘ－０６／５６（ＴＴＬレベルＰＩＯボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.7-1　ADSP32X-06/56 ドライバウィンドウ



この中には、ＡＤＳＰ３２Ｘ－０６／５６とのインターフェースを取るための４つのデバイスドライバブロックが登録されています。

- TTL bit input　　ビット単位の入力デバイスドライバです。
- TTL port input　　ポート単位の入力デバイスドライバです。
- TTL bit output　　ビット単位の出力デバイスドライバです。
- TTL port output　　ポート単位の出力デバイスドライバです。
- TTL paralell input　複数ビットの同時入力デバイスドライバです。
- TTL paralell output　複数ビットの同時出力デバイスドライバです。
- TTL bit output Terminate output
  終了時の出力指定が行える、ビット単位の出力デバイスドライバです。
- TTL port output Terminate output
  終了時の出力指定が行える、ポート単位の出力デバイスドライバです。

ビット単位のデバイスドライバは、任意のポートの任意のビットをランダムにアクセスできます。外部機器で２値化されたデータ、操作スイッチ、リミットセンサー等のデジタル信号を取り扱うことができます。

ポート単位のデバイスドライバは、ポートを３２ビット符号付き整数として取り扱います。データ形式はストレートバイナリで、-2^{31} ～ $2^{31}-1$の範囲のデータとなります。外部機器でデジタル化されたデータを取り扱うことができます。

複数ビットのデバイスドライバは、複数のビットを同時に入出力します。ビット単位のデバイスドライバーを複数並べ、Mux（Demux）にてベクトル／スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバーの方がビット単位のデバイスドライバーを複数並べるより高速に入出力が行えます。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、ＵＩＬ、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

2.7.2　ポート／ビット番号とボードとの対応

ＡＤＳＰ３２Ｘ－０６／５６は最大４台使用できます。ＡＤＳＰ３２Ｘ－０６／５６にはＩ／Ｏポートが１ポートあり、各デバイスドライバブロックはこれらポートに論理ポート番号（論理ビット番号）を割り付けて取り扱います。ＡＤＳＰ３２Ｘ－０６／５６の各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表 2.7.1　及び　表 2.7.2　の通りです。

ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表 2.7.1　ポート番号とＡＤＳＰ３２Ｘ－０６／５６ボードとの対応

ボード	ポート番号
１台目	0
２台目	1
３台目	2
４台目	3

表 2.7.2　ビット番号とＡＤＳＰ３２Ｘ－０６／５６ボードとの対応

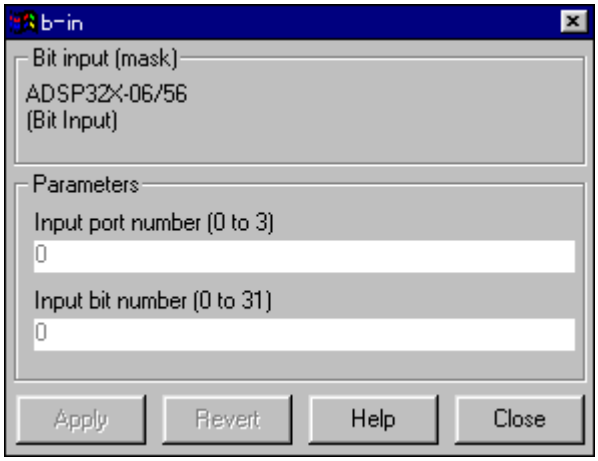
入出力信号名	ビット番号	ポート単位
ID/OD31	３１	符号
ID/OD30	３０	2^{30}
ID/OD29	２９	2^{29}
ID/OD28	２８	2^{28}
～	～	～
ID/OD03	3	2^{3}
ID/OD02	2	2^{2}
ID/OD01	1	2^{1}
ID/OD00	0	2^{0}

注：各ポート共通

2.7.3　ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時０、HIGH の時１です。入力信号が接続されていないビットは１になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.7-2　ADSP32X-06/56 TTL bit input



- Input port number (0 to 3)

  入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (43ページ) を参照してください。

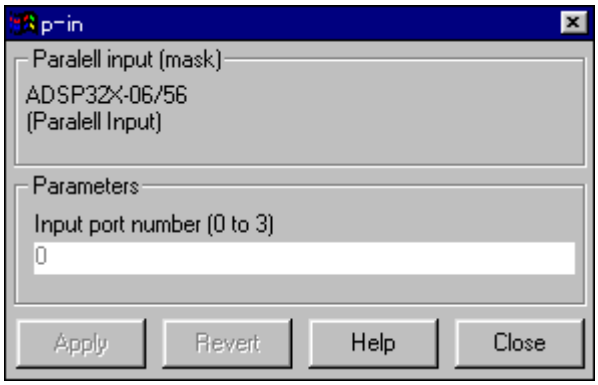
- Input bit number (0 to 31)

  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2 (44ページ) を参照してください。

2.7.4　ポート単位入力デバイスドライバブロック

ポート単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１とした、符号付き３２ビット整数とみなした値です。
ボードが実装されていないポートの値は不定です。

図 2.7-3　ADSP32X-06/56 TTL port input



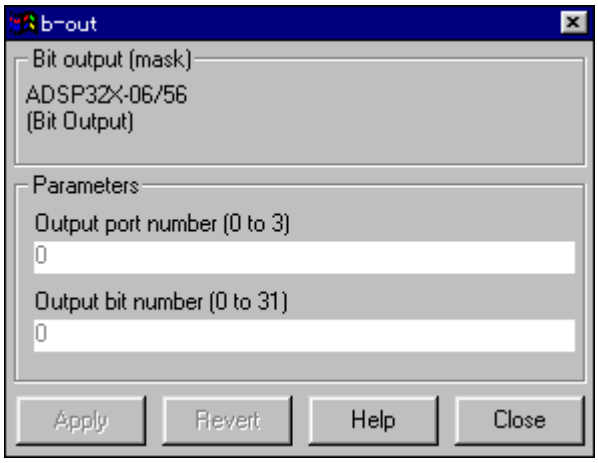
- Input port number (0 to 3)

  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1（43ページ）を参照してください。

2.7.5　ビット単位出力デバイスドライバブロック

ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 2.7-4　ADSP32X-06/56 TTL bit output



- Output port number (0 to 3)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1（43ページ）を参照してください。

- Output bit number (0 to 31)

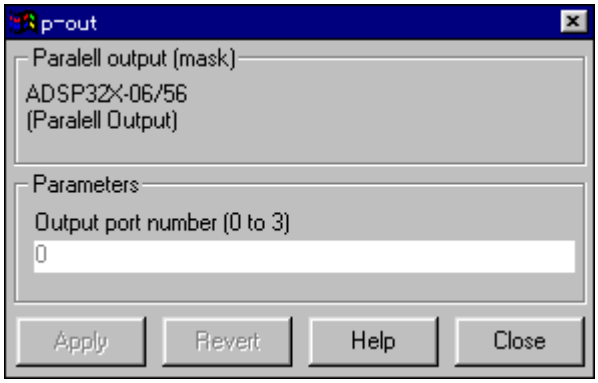
  出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2（44ページ）を参照してください。

注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

2.7.6　ポート単位出力デバイスドライバブロック

ポート単位出力デバイスドライバブロックは、スカラーの入力を受取り、符号付き３２ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時にLOW、１の時HIGHです。

図 2.7-5　ADSP32X-06/56 TTL port output



- Output port number (0 to 3)

  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

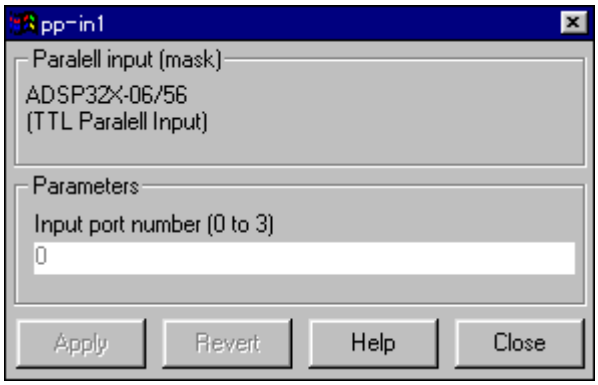
2.7.7　複数ビット入力デバイスドライバブロック

複数ビット入力デバイスドライバブロックは、指定されたポートを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは幅３２のベクトルです。ベクトルの先頭の要素がビット番号０、次がビット番号１の順で、最後の要素がビット番号３１に対応します。ベクトルの各要素の値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１です。

ポート入力デバイスドライバはとの相違点は、ポート入力はＴＴＬ入力の１ポートを１つのデータと見なすのに対し、このブロックは１つ１つのビットの状態が個々の信号（０か１）として入力できます。

ボードが実装されていない場合の値は不定です。

図 2.7-6　ADSP32X-06/56 TTL paralell input



- Input port number (0 to 3)

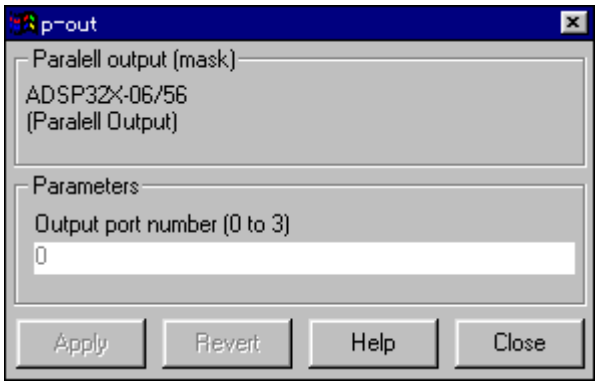
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1（43ページ）を参照してください。

2.7.8　複数ビット出力デバイスドライバブロック

複数ビット出力デバイスドライバブロックは、ベクトルの入力を受取り、各要素に対応するビットにＨＩＧＨ又はＬＯＷを出力します。各要素の値が０の時にはＬＯＷを、０以外の時にはＨＩＧＨを出力します。このブロックへの入力は幅３２のベクトルを与えます。ベクトルの先頭の要素がビット番号０、次がビット番号１の順で、最後の要素がビット番号３１に対応します。

ポート出力デバイスドライバはとの相違点は、ポート出力はそのブロックへの入力をバイナリに変換し、各ビットに対応するＴＴＬポートのビットにＨＩＧＨ／ＬＯＷを出力しますが、このブロックは入力ベクトルの個々の要素の状態がＴＴＬポートの個々のビットにＨＩＧＨ／ＬＯＷで出力されます。

図 2.7-7　ADSP32X-06/56 TTL port output



- Output port number (0 to 3)

  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

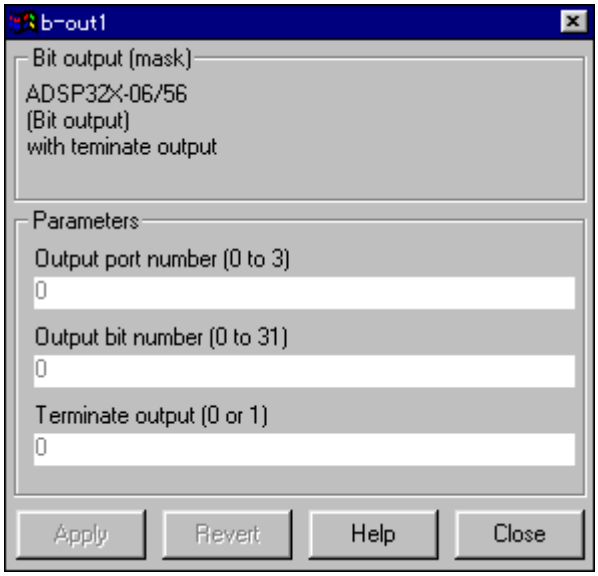
注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.7.9　終了時出力指定 ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 2.7-8　ADSP32X-06/56 TTL bit output Terminate output



- Output port number (0 to 3)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1（43ページ）を参照してください。

- Output bit number (0 to 31)

  出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2（44ページ）を参照してください。

- Terminate output　(0 or 1)

  実時間モデルが停止する際に出力して欲しい状態を設定します。０でＬＯＷ，１でＨＩＧＨが出力されます。

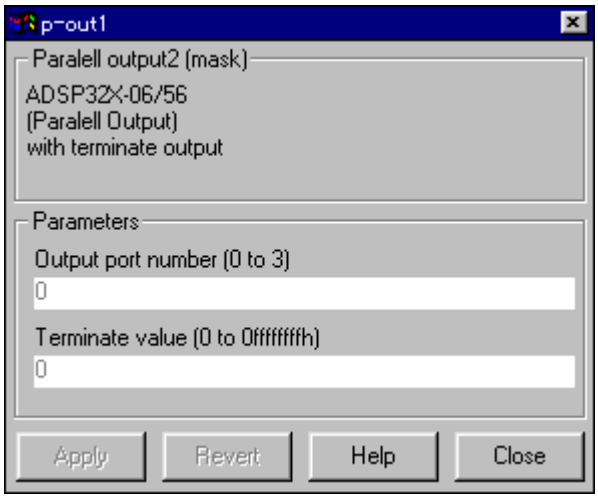
注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

2.7.10 終了時出力指定 ポート単位出力デバイスドライバブロック

終了時出力指定付きポート単位出力デバイスドライバブロックは、スカラーの入力を受取り、符号付き３２ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時にLOW、１の時HIGHです。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.7-9 ADSP32X-06/56 TTL port output Terminate output



- Output port number (0 to 3)

  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

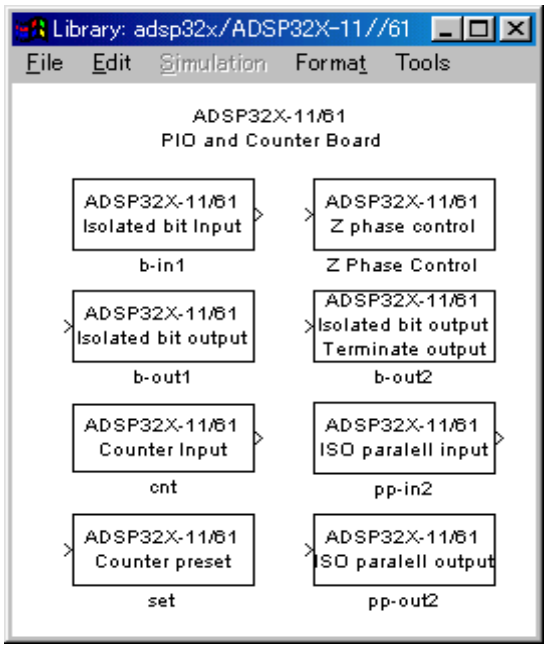
- Terminate value (0 to 0ffffffffh)

  実時間モデルが停止する際に出力して欲しい値を１６進数で設定します。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

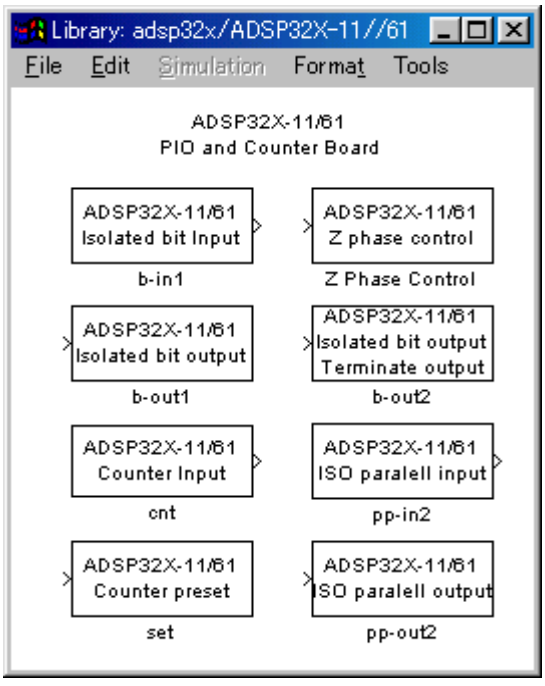
2.8 ＡＤＳＰ３２Ｘ－１１／６１デバイスドライバーブロック

2.8.1　　概要



に、 ＡＤＳＰ３２Ｘ－１１／６１（ＰＩＯ&カウンタボード）用デバイスドライバーウィンドウを開いた所を示します。

図 2.8-1　ＡＤＳＰ３２Ｘ－１１／６１ドライバウィンドウ



この中には、ＡＤＳＰ３２Ｘ－１１／６１とのインターフェースを取るためのデバイスドライバブロックが登録されています。

- Isolated bit input　ビット単位の絶縁入力デバイスドライバブロックです。
- Isolated bit output ビット単位の絶縁出力デバイスドライバブロックです。
- Counter input　　プリセット付きカウンターデバイスドライバです。
- Counter preset　　カウンタープリセットデバイスドライバブロックです。
- Z phase control　　カウンタリセットデバイスドライバブロックです。
- Isolated bit output Teminate output
  終了時の出力指定が行えるビット単位の絶縁出力デバイスドライバブロックです。
- Isolated paralell input
  複数ビット同時の絶縁入力デバイスドライバブロックです。
- Isolated paralell output
  複数ビット同時の絶縁出力デバイスドライバブロックです。

ビット単位の入出力デバイスドライバは、ＡＤＳＰ３２Ｘ－１１／６１ボードの任意の絶縁入出力ポートの任意のビットに信号を入出力します。アクセスは全てビット単位で行なわれます。

プリセット付きカウンターデバイスドライバは、カウンターをアクセスするデバイスドライバで、初期化時にカウンタを規定の値にセットすることもできます。

カウンタプリセットデバイスドライバは、任意のカウンタを、任意の時刻にプリセットするドライバです。

カウンタリセットデバイスドライバは、カウンタを外部からリセットするドライバです。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、ＵＩＬ、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

複数ビットのデバイスドライバは、複数のビットを同時に入出力します。ビット単位のデバイスドライバーを複数並べ、Mux（Demux）にてベクトル／スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバーの方がビット単位のデバイスドライバーを複数並べるより高速に入出力が行えます。

2.8.2　ビット番号、ポート／チャンネル番号とボードとの対応

ＡＤＳＰ３２Ｘ－１１／６１は最大４台使用できます。ＡＤＳＰ３２Ｘ－１１／６１ボードにはカウンターが４チャンネルと絶縁Ｉ／Ｏポートが１ポート実装されています。ＡＤＳＰ３２Ｘ－１１／６１の各デバイスドライバブロックは、これらのポートを０から始まる連続する論理番号で取り扱います。各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表.2.8. 1　及び　表.2.8. 2　の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表.2.8. 1　ポート／チャンネル番号とＡＤＳＰ３２Ｘ－１１／６１との対応

ボード	カウンタチャンネル番号				絶縁ポート番号
	ｶｳﾝﾀｰ０	ｶｳﾝﾀｰ１	ｶｳﾝﾀｰ２	ｶｳﾝﾀｰ３	
１台目	０	１	２	３	０
２台目	４	５	６	７	１
３台目	８	９	１０	１１	２
４台目	１２	１３	１４	１５	３

表.2.8. 2　ビット番号とＡＤＳＰ３２Ｘ－１１／６１との対応

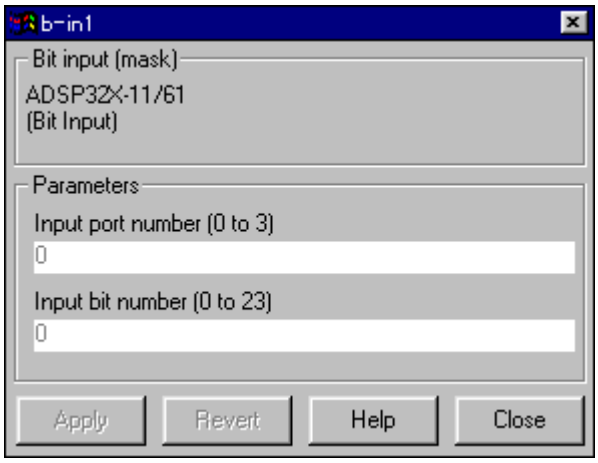
ビット番号	絶縁入力信号名	絶縁出力信号名
0	I0_00	00_00
1	I0_01	00_01
2	I0_02	00_02
3	I0_03	00_03
4	I0_04	00_04
5	I0_05	00_05
6	I0_06	00_06
7	I0_07	00_07
8	I0_10	00_10
9	I0_11	00_11
１０	I0_12	00_12
１１	I0_13	00_13
１２	I0_14	00_14
１３	I0_15	00_15
１４	I0_16	00_16
１５	I0_17	00_17
１６	I0_20	－
１７	I0_21	－
１８	I0_22	－
１９	I0_23	－
２０	I0_24	－
２１	I0_25	－
２２	I0_26	－
２３	I0_27	－

注１：各絶縁入出力ポート共通

2.8.3　ビット単位、絶縁入力デバイスドライバブロック

ビット単位絶縁入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.8-2　ADSP32X-11/61 Isolated bit input



- Input port number (0 to 3)

  入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

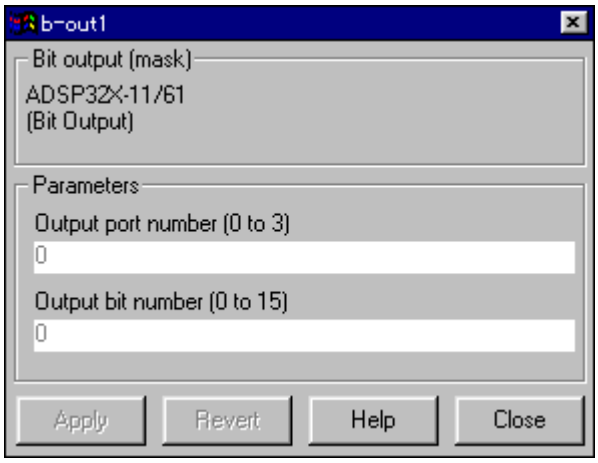
- Input bit number (0 to 23)

  入力の対象となるビット番号を指定します。指定できるビット番号は０～２３です。ビット番号とハードウェアとの対応は、表.2.8. 2（56ページ）を参照してください。

2.8.4　ビット単位、絶縁出力デバイスドライバブロック

ビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが０の時に OFF、１の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。

図 2.8-3　ADSP32X-11/61 Isolated bit output



- Output port number (0 to 3)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

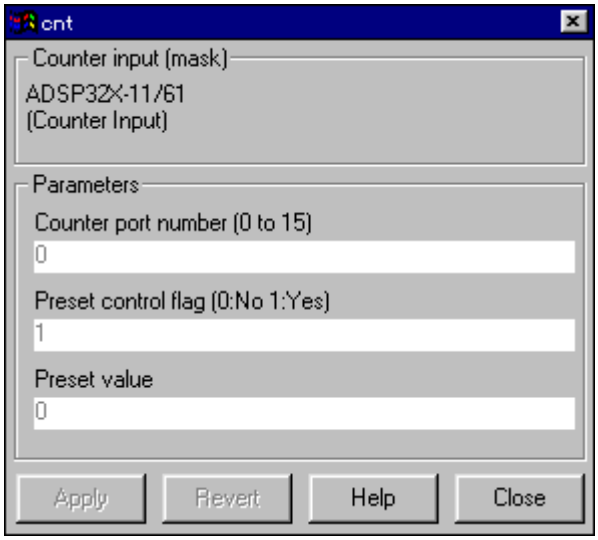
- Output bit number (0 to 15)

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表.2.8. 2（56ページ）を参照してください。

2.8.5　プリセット付きカウンタデバイスドライバブロック

プリセット付きカウンタデバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。このブロックの出力はスカラーです。ADSP32X-11/61 ボードの指定されたカウンターは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。

図 2.8-4　ADSP32X-11/61 Counter input



- Counter port number (0 to 15)

  入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

- Preset controll flagg (0:No 1:Yes)

  初期化時にカウンタ値のプリセットを行なうか否かを決定します。０又は１を設定してください。０を設定した場合はプリセットは行ないません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。

- Preset value

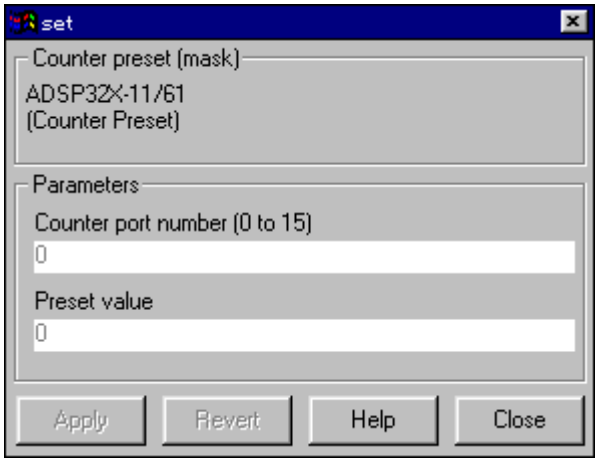
  Preset controll flagg に１を設定した場合、ここに設定した値がカウンターの初期値として採用されます。 Preset controll flagg に１を設定した場合のみ有効です。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

2.8.6　カウンタプリセットデバイスドライバブロック

カウンタプリセットデバイスドライバブロックは、任意のチャンネル番号のカウンターを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に１度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。

このブロックには１つのスカラー入力があり、ここに０又は１の値を入力することによりプリセットタイミングを制御します。０が入力されている期間はカウンタはカウント動作を行ないます。１が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図 2.8-5　ADSP32X-11/61 Counter preset



- Counter port number (0 to 15)

  入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

- Preset value

  ここに設定した値がカウンターにセットされます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（－2^{31} ～ 2^{31}－１）です。

2.8.7　カウンタリセットデバイスドライバブロック

カウンタリセットデバイスドライバブロックは、任意のポート番号のカウンターを外部からのリセットを許可するデバイスドライバです。

このブロックに、１６のベクトル入力があり、ここに各ポートへのリセット許可を設定します。０はリセット無許可、１は許可です。このブロックはパラメータはありません。ベクトルの１番目よりカウンタポート番号０番に対応しています。

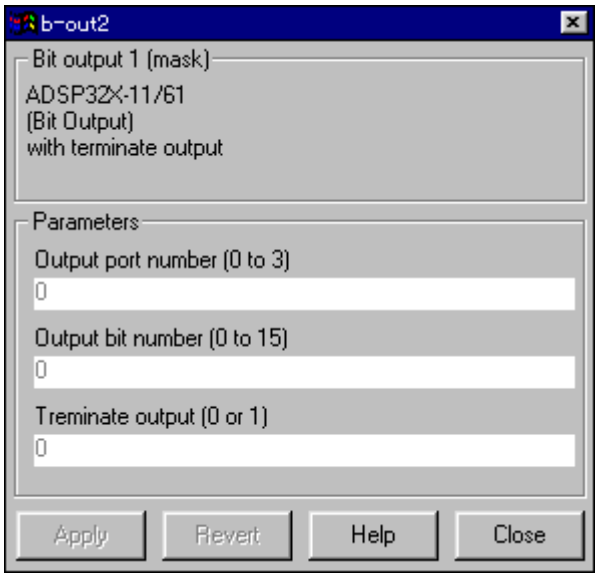
外部信号については、ADSP32X-11/61 ハードウェアマニュアル“6.3 カウンターの外部からのリセット“を参照してください。

2.8.8　終了時出力指定 ビット単位、絶縁出力デバイスドライバブロック

終了時出力指定付きビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが０の時に OFF、１の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。

このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力する事ができます。

図 2.8-6　ADSP32X-11/61 Isolated bit output Terminate output



- Output port number (0 to 3)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

- Output bit number (0 to 15)

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表.2.8. 2（56ページ）を参照してください。

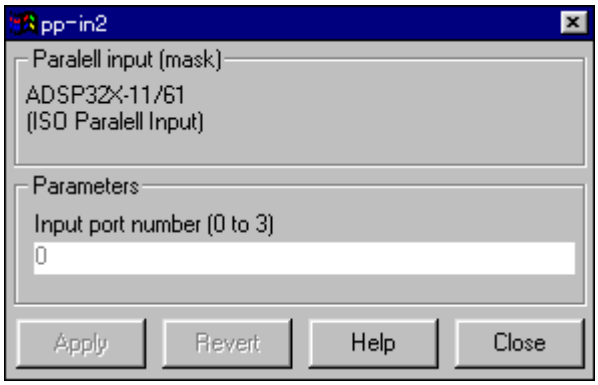
- Terminale output (0 or 1)

  実時間モデルが終了する際に出力したい値を設定します。

2.8.9　複数ビット同時、絶縁入力デバイスドライバブロック

複数ビット同時絶縁入力デバイスドライバブロックは、指定されたポートの全ビットを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは、幅が２４のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット２３に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.8-7　ADSP32X-11/61 ISO paralell input



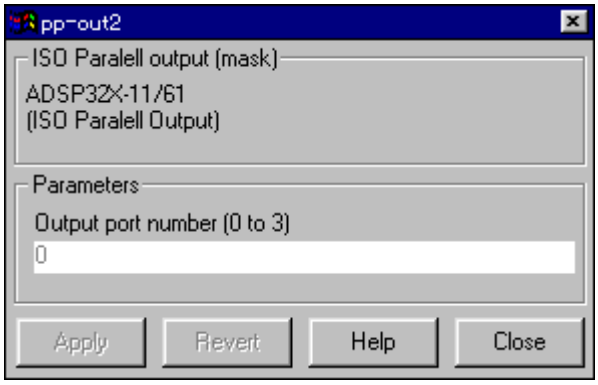
- Input port number (0 to 3)

  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1 (55ページ) を参照してください。

2.8.10 複数ビット同時、絶縁出力デバイスドライバブロック

複数ビット同時絶縁出力デバイスドライバブロックは、ベクトルの入力を受取り、指定されたポートの各ビットへ出力します。入力するベクトルは幅１６のベクトルで、先頭の要素が出力ビット０、次が出力ビット１の順で、最後の要素が出力ビット１５に相当します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力が０の時に OFF、０以外の時 ON です。

図 2.8-8　ADSP32X-11/61 ISO paralell output



- Output port number (0 to 3)

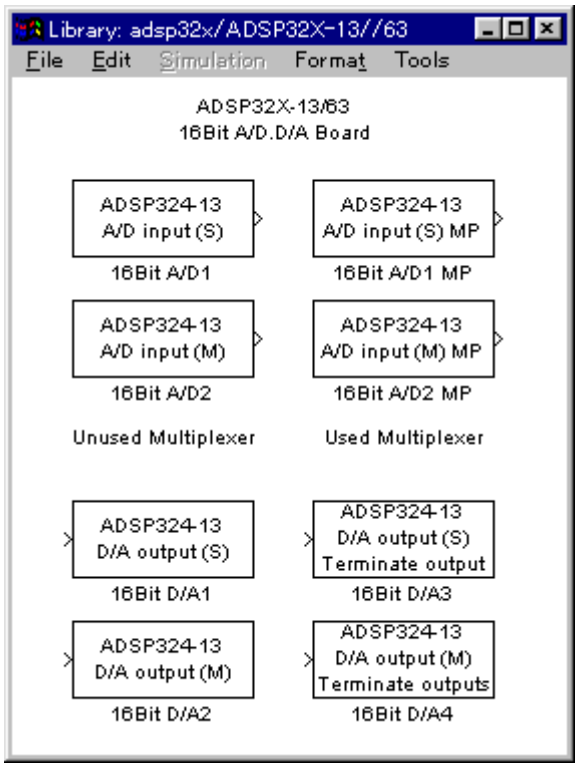
  出力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表.2.8. 1（55ページ）を参照してください。

2.9　ＡＤＳＰ３２Ｘ－１３／６３デバイスドライバーブロック

2.9.1　概要

図 2.9-1に、ＡＤＳＰ３２Ｘ－１３／６３（１６ビットＡ／Ｄ・Ｄ／Ａボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.9-1　ADSP32X-13/63 ドライバウィンドウ



この中にはＡＤＳＰ３２Ｘ－１３／６３とのインターフェースを取るためのデバイスドライバーブロックが登録されています。

- A/D input (S)　　１チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
  マルチプレクサー（オプション）が無い場合に利用出来ます。
- A/D input (M)　　多チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
  マルチプレクサー（オプション）が無い場合に利用出来ます。
- A/D input (S) MP　　１チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
  マルチプレクサー（オプション）が有る場合に利用出来ます。
- A/D input (M) MP　　多チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
  マルチプレクサー（オプション）が有る場合に利用出来ます。
- D/A input (S)　　１チャンネルのＤ／Ａ出力デバイスドライバブロック

です。

- D/A input (M)　　多チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (S)　　Terminate output
  終了時出力指定付きの１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時出力指定付きの多チャンネルのＤ／Ａ出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、ＵＩＬ、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのＡ／Ｄ入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。Ｄ／Ａの場合も同様です。特に、Ａ／Ｄでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.9.2　チャンネル番号とボードとの対応

ＡＤＳＰ３２Ｘ－１３／６３ボードは最大４台使用できます。ＡＤＳＰ３２Ｘ－１３／６３ボードにはＡ／Ｄが２チャンネル（マルチプレクサーオプションを増設した場合は１６チャンネル）と、Ｄ／Ａが２チャンネル実装されています。ＡＤＳＰ３２Ｘ－１３／６３の各デバイスドライバブロックは、これらのチャンネルを０から始まる連続する論理チャンネル番号として取り扱います。ＡＤＳＰ３２Ｘ－１３／６３の各デバイスドライバブロックが取り扱うチャンネル番号とハードウェアとの対応は、表 2.9.1 及び 表 2.9.2の通りです。

ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

チャンネル番号とＡＤＳＰ３２Ｘ－１３／６３ボードとの対応

表 2.9.1　マルチプレクサが無い場合

ボード	信号名	
	IN/OUT00	IN/OUT01
１台目	０	１
２台目	２	３
３台目	４	５
４台目	６	７

表 2.9.2　マルチプレクサが有る場合

ボード	信号名							
	IN00	IN01	…	IN07	IN10	IN11	…	IN17
１台目	０	１	…	７	８	９	…	１５
２台目	１６	１７	…	２３	２４	２５	…	３１
３台目	３２	３３	…	３９	４０	４１	…	４７
４台目	４８	４９	…	５５	５６	５７	…	６３

2.9.3　ＰＧＡバイパス

マルチプレクサーオプション（ADSP324-12）を追加した場合、Ａ／Ｄ変換は、マルチプレクサーを切り替えながら行う為、変換に要する時間が長くなります。これは、Ａ／Ｄチャンネルの切り替えの際に、ボードに実装されている ＰＧＡ（Programable Gain Amp）やＳ／Ｈ（Sample & Hold）の応答時間（セトリングタイム）を待機している為です。この時間の大半はＰＧＡが占めているため、ＰＧＡを取り除く（バイパスさせる）事により大幅に短縮できます。（約３０４μ秒）

ＰＧＡをバイパスするには、以下の点の注意が必要です。

（１）入力がシングルエンドになります。（標準は差動入力です。）

（２）ハードウェアの改造が必要です。

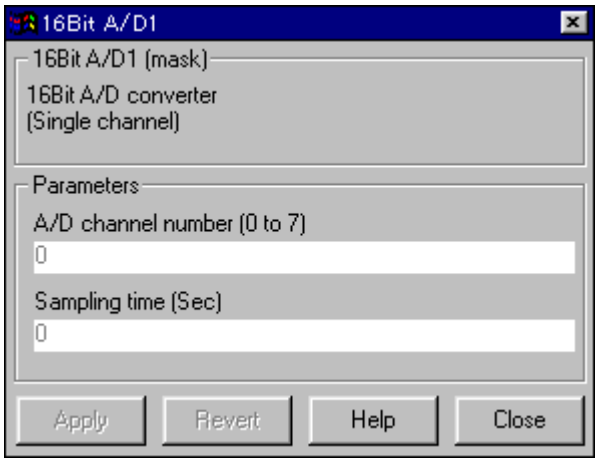
（３）ハードウエアの改造を行わずに、ブロックの PGA amp パラメータを０にすると、正しい値が取り込めません。

詳細については、メーカーにお問い合わせください。

2.9.4　１チャンネルＡ／Ｄ入力デバイスドライバブロック

１チャンネルＡ／Ｄ入力デバイスドライバブロックは、指定されたチャンネル番号のＡ／Ｄからデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。
このデバイスドライバーはマルチプレクサー（オプション）が無い場合のみ利用可能です。

図 2.9-2　ADSP32X-13/63 A/D input (S)



- A/D channel number (0 to 7)

  入力するＡ／Ｄのチャンネル番号を０～７の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1（67ページ）を参照してください。

- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

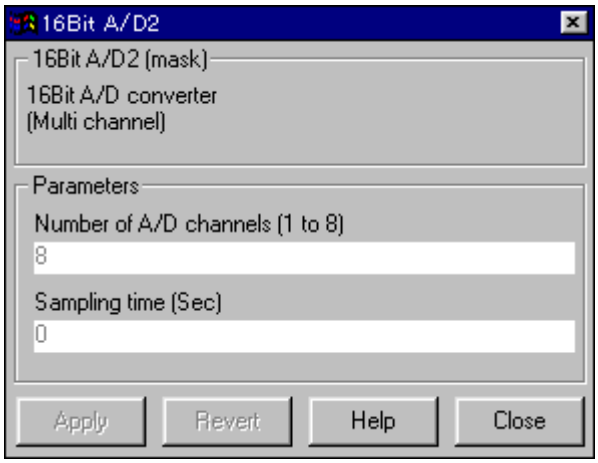
2.9.5　多チャンネルＡ／Ｄ入力デバイスドライバブロック

多チャンネルＡ／Ｄ入力デバイスドライバブロックは、複数チャンネルのＡ／Ｄから一括してデータを入力します。入力の対象となるＡ／Ｄチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。

このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバーはマルチプレクサー（オプション）が無い場合のみ利用可能です。

図 2.9-3　ADSP32X-13/63 A/D input (M)



- Number of A/D channels (1 to 8)

  入力するＡ／Ｄのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1（67ページ）を参照してください。

- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

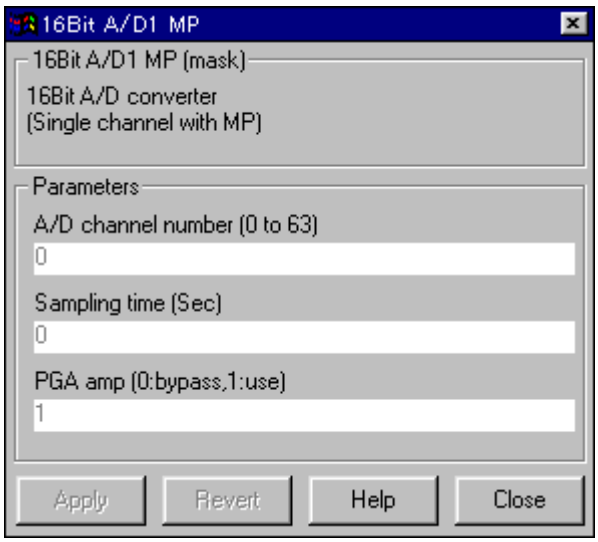
2.9.6　１チャンネルＡ／Ｄ入力（ＭＰ付き）デバイスドライバブロック

１チャンネルＡ／Ｄ入力（ＭＰ＝マルチプレクサ付き）デバイスドライバブロックは、指定されたチャンネル番号のＡ／Ｄからデータを入力します。このブロックからの出力はスカラーとなります。

このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバーはマルチプレクサー（オプション）が有る場合のみ利用可能です。

図 2.9-4　ADSP32X-13/63 A/D input (S)　MP



- A/D channel number (0 to 63)

  入力するＡ／Ｄのチャンネル番号を０～６３の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.2（67ページ）を参照してください。

- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

- PGA amp (0:bypass,1:use)

  ボード上のＰＧＡを使用しているか否を設定します。通常は１を設定してください。ＡＤ変換の高速化の為、ＰＧＡのバイパス（2.9.3参照）を実施した場合のみ０としてください。

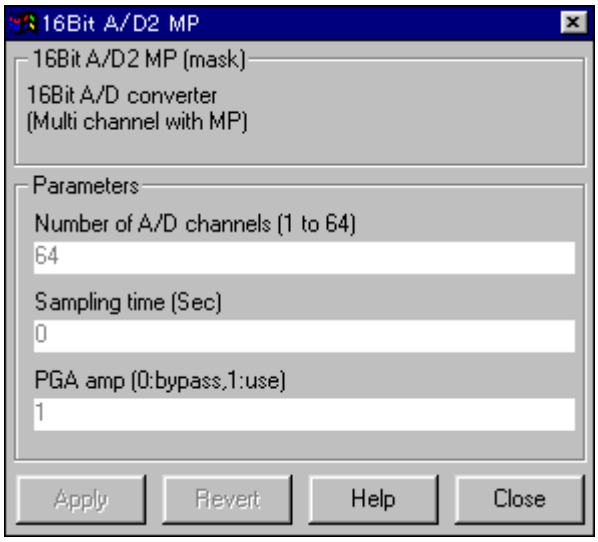
2.9.7　多チャンネルＡ／Ｄ入力（ＭＰ付き）デバイスドライバブロック

多チャンネルＡ／Ｄ入力（ＭＰ＝マルチプレクサ付き）デバイスドライバブロックは、複数チャンネルのＡ／Ｄから一括してデータを入力します。入力の対象となるＡ／Ｄチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。

このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバーはマルチプレクサー（オプション）が有る場合のみ利用可能です。

図 2.9-5　ADSP32X-13/63 A/D input (M) MP



- Number of A/D channels (1 to 64)

  入力するＡ／Ｄのチャンネル数を１～６４の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.2（67ページ）を参照してください。

- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

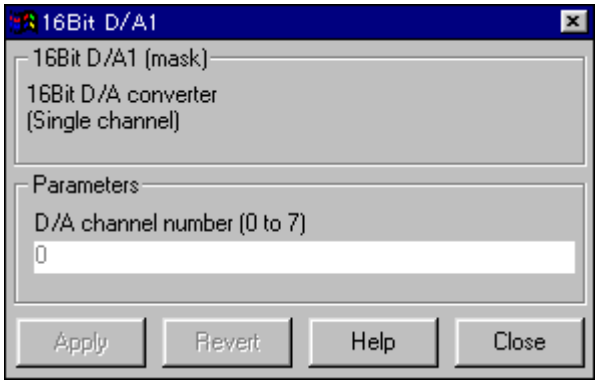
- PGA amp (0:bypass,1:use)

  ボード上のＰＧＡを使用しているか否を設定します。通常は１を設定してください。ＡＤ変換の高速化の為、ＰＧＡのバイパス（2.9.3参照）を実施した場合のみ０としてください。

2.9.8　１チャンネルＤ／Ａ出力デバイスドライバブロック

１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-6　ADSP32X-13/63 D/A output (S)



- D/A channel number (0 to 7)

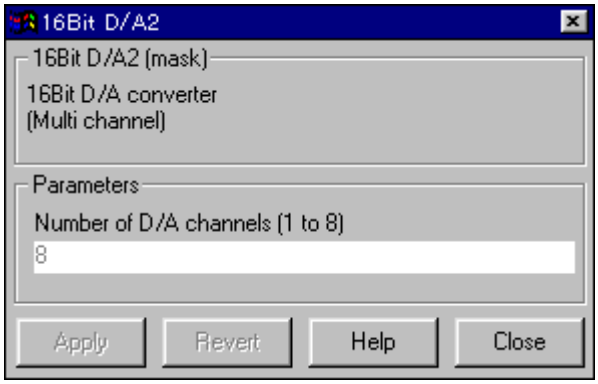
  出力するＤ／Ａのチャンネル番号を０～７の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1（67ページ）を参照してください。

2.9.9　多チャンネルD／A出力デバイスドライバブロック

多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。出力の対象となるD／Aチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへの入力ベクトルの最初の要素がチャンネル番号０のD／Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-7　ADSP32X-13/63 D/A output (M)



- Number of D/A channels (1 to 8)

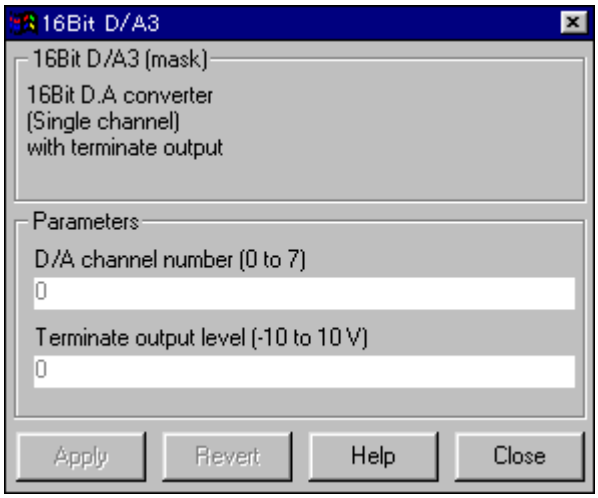
  出力するD／Aのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (67ページ) を参照してください。

2.9.10 終了時出力指定　１チャンネルＤ／Ａ出力デバイスドライバブロック

終了時出力指定付き１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-8　ADSP32X-13/63 D/A output (S) Terminate output



- D/A channel number (0 to 7)

  出力するＤ／Ａのチャンネル番号を０～７の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (67ページ) を参照してください。

- Terminate output data (-10 to 10V)

  リアルタイムモデルが終了する際にＤ／Ａから出力する値を設定します。単位は[Volt]です。

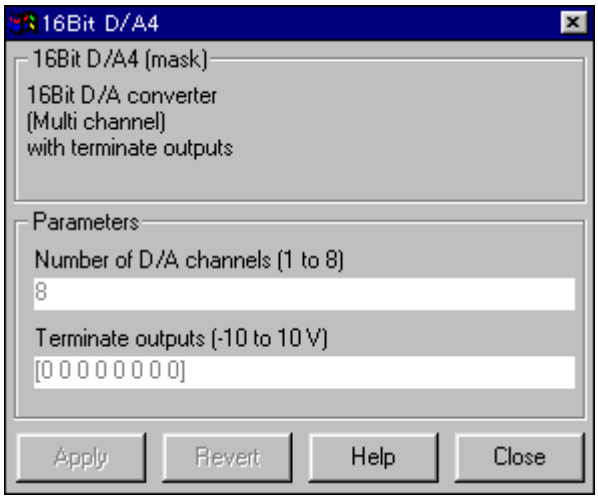
2.9.11 終了時出力指定 多チャンネルD／A出力デバイスドライバブロック

終了時出力指定付き 多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。出力の対象となるD／Aチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへの入力ベクトルの最初の要素がチャンネル番号０のD／Aへ出力されます。

このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-9 ADSP32X-13/63 D/A output (M) Terminate output



- Number of D/A channels (1 to 8)

  出力するD／Aのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (67ページ) を参照してください。

- Terminate output data (-10 to 10V)

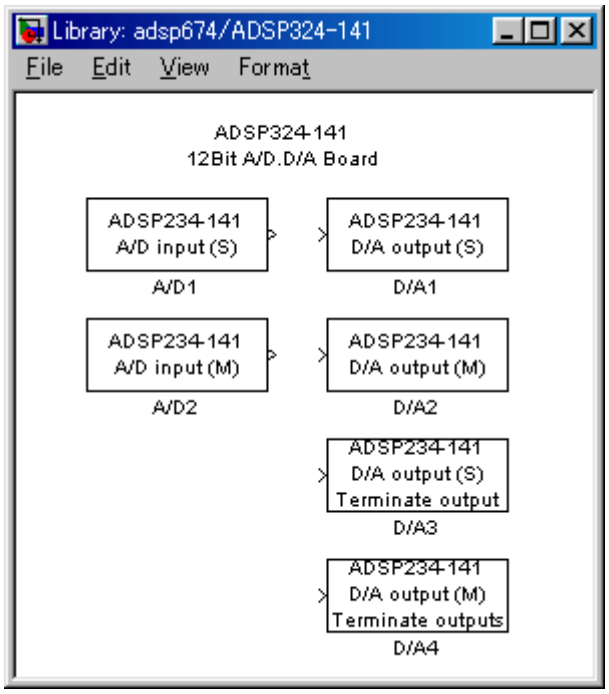
  リアルタイムモデルが終了する際にD／Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.10ＡＤＳＰ３２４－１４１デバイスドライバーブロック

2.10.1 概要

図 2.10-1に、ＡＤＳＰ３２４－１４１（１２ビットＡ／Ｄ・Ｄ／Ａボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.10-1　ADSP324-141 ドライバウィンドウ



この中にはＡＤＳＰ３２４－１４１とのインターフェースを取るための６つのデバイスドライバーブロックが登録されています。

- A/D input (S)　　１チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
- A/D input (M)　　多チャンネルのＡ／Ｄ入力デバイスドライバブロックです。
- D/A input (S)　　１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M)　　多チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (S) Terminate output
  終了時の出力指定が行える、１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時の出力指定が行える、多チャンネルのＤ／Ａ出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、ＵＩＬ、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのＡ／Ｄ入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。Ｄ／Ａの場合も同様です。特に、Ａ／Ｄでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.10.2 チャンネル番号とボードとの対応

ＡＤＳＰ３２４－１４１は最大４台使用でき、１台のボードにＡ／Ｄ、Ｄ／Ａが各１６及び８チャンネル（オプション実装時１２チャンネル）実装されています。ＡＤＳＰ３２４－１４１の各デバイスドライバブロックはこれらのチャンネルを連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 2.10.1 及び の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表 2.10.1　入力チャンネル番号とＡＤＳＰ３２４－１４１ボードとの対応

ボード	IN0	IN1	…	IN14	IN15
１台目	０	１	…	１４	１５
２台目	１６	１７	…	３０	３１
３台目	３２	３３	…	４６	４７
４台目	４８	４９	…	６２	６３

表 2.10.2　出力チャンネル番号とＡＤＳＰ３２４－１４１ボードとの対応

(D/A 標準実装の場合)

ボード	OUT0	OUT1	…	OUT6	OUT7
１台目	０	１	…	６	７
２台目	８	９	…	１４	１５
３台目	１６	１７	…	２２	２３
４台目	２４	２６	…	３０	３１

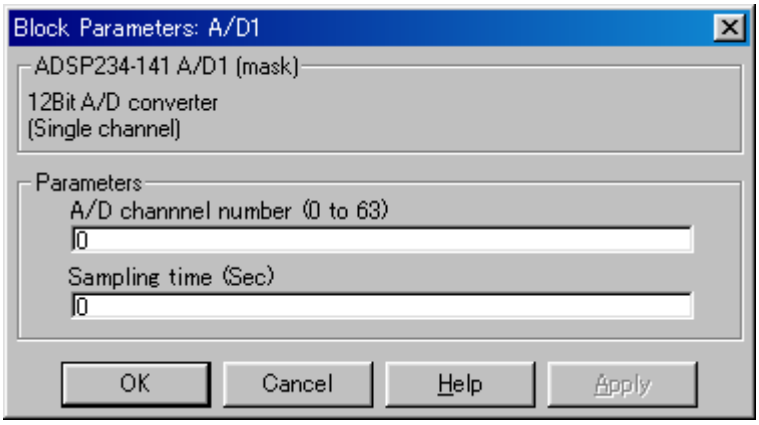
(D/A の CH 増設オプション有りの場合)

ボード	OUT0	OUT1	…	OUT10	OUT11
１台目	０	１	…	１０	１１
２台目	１２	１３	…	２２	２３
３台目	２４	２５	…	３４	３５
４台目	３６	３７	…	４６	４７

2.10.3 １チャンネルＡ／Ｄ入力デバイスドライバブロック

１チャンネルＡ／Ｄ入力デバイスドライバブロックは、指定されたチャンネル番号のＡ／Ｄからデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.10-2　ADSP324-141 A/D input (S)



- A/D channel number (0 to 63)

  入力するＡ／Ｄのチャンネル番号を０～６３の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.1 (79ページ) を参照してください。

- Sampling time (Sec)

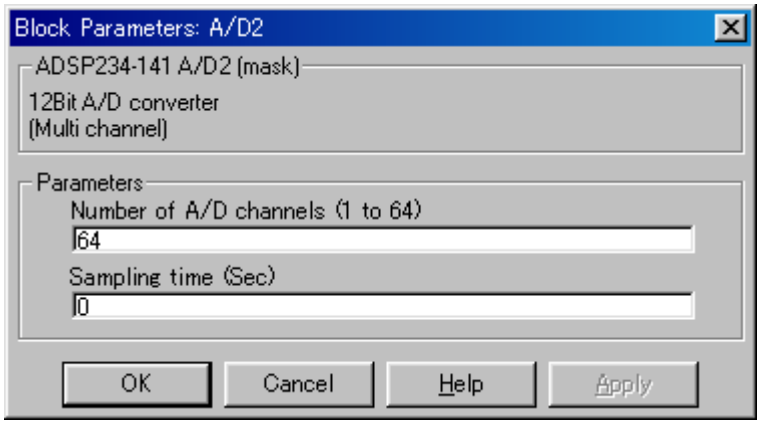
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

2.10.4 多チャンネルＡ／Ｄ入力デバイスドライバブロック

多チャンネルＡ／Ｄ入力デバイスドライバブロックは、複数チャンネルのＡ／Ｄから一括してデータを入力します。入力の対象となるＡ／Ｄチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。

このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.10-3　ADSP324-141 A/D input (M)



- Number of A/D channels (1 to 64)

  入力するＡ／Ｄのチャンネル数を１～６４の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.1（79ページ）を参照してください。

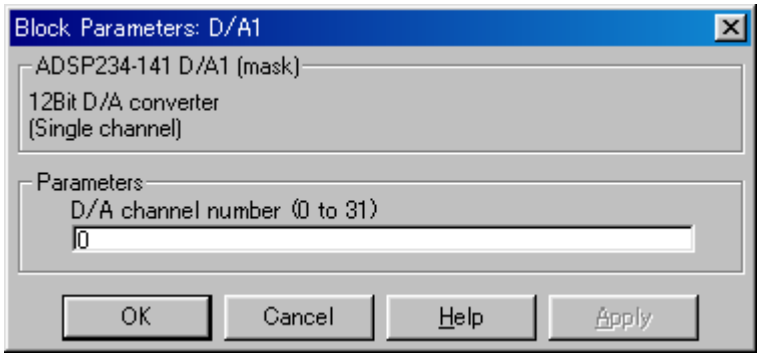
- Sampling time (Sec)

  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

2.10.5 １チャンネルＤ／Ａ出力デバイスドライバブロック

１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-4　ADSP324-141 D/A output (S)



- D/A channel number (0 to 31)

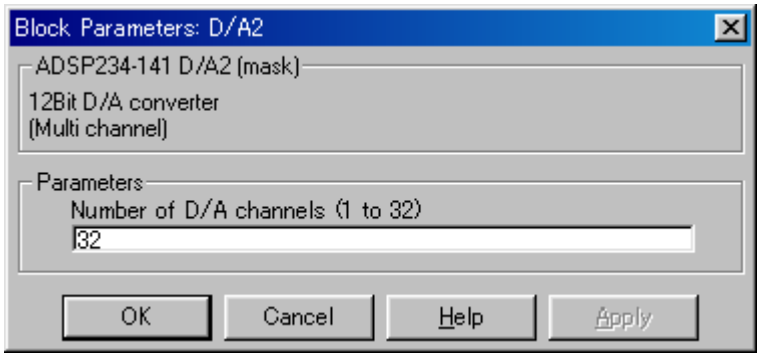
  出力するＤ／Ａのチャンネル番号を０～３１の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2（79ページ）を参照してください。

2.10.6 多チャンネルD／A出力デバイスドライバブロック

多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。出力の対象となるD／Aチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のD／Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-5　ADSP324-141 D/A output (M)



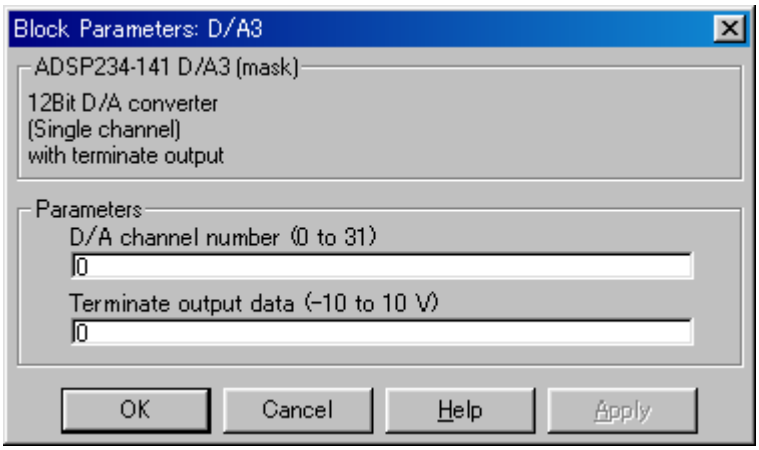
- Number of A/D channels (1 to 32)

  出力するD／Aのチャンネル数を１～３２の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2（79ページ）を参照してください。

2.10.7 終了時出力指定１チャンネルＤ／Ａ出力デバイスドライバブロック

終了時出力指定付き１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-6　ADSP324-141 output (S) Terminate output



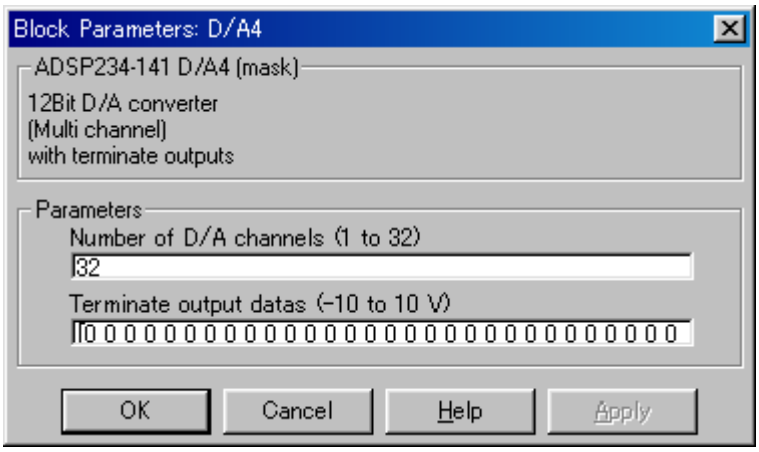
- D/A channel number (0 to 31)
  出力するＤ／Ａのチャンネル番号を０～３１の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2 (79ページ) を参照してください。
- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際にＤ／Ａから出力する値を設定します。単位は[Volt]です。

2.10.8 終了時出力指定 多チャンネルD／A出力デバイスドライバブロック

終了時出力指定付き多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD／Aへ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-7　ADSP324-141 D/A output (M) Terminate output



- Number of D/A channels (1 to 32)

  出力するD／Aのチャンネル数を１～３２の整数で指定します。先頭チャンネル番号は0で固定です。チャンネル番号とハードウェアとの対応は、表 2.10.2 (79ページ) を参照してください。

- Terminate output data (-10 to 10V)

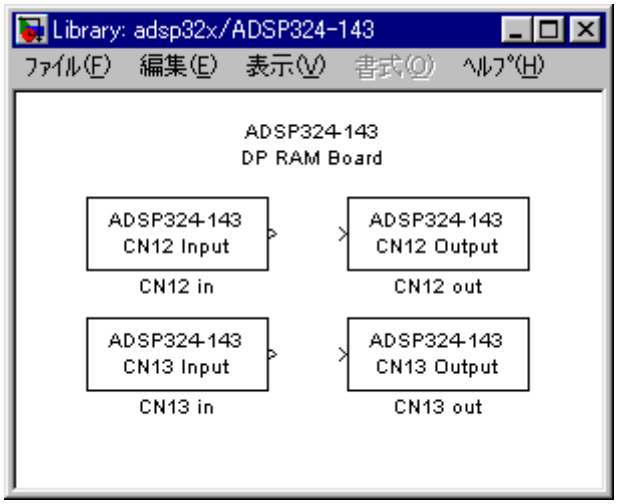
  リアルタイムモデルが終了する際にD／Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.11ＡＤＳＰ３２４－１４３デバイスドライバーブロック

2.11.1 概要

図 2.11-1に、ＡＤＳＰ３２４－１４３（ＤＰ　ＲＡＭボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.11-1　ADSP324-143 ドライバウィンドウ



この中にはＡＤＳＰ３２４－１４３とのインターフェースを取るための４つのデバイスドライバーブロックが登録されています。

- CN12 input　　前段 DSP との間にあるＤＰＲＡＭを使い、前段 DSP からデータを受け取るデバイスドライバブロックです。
- CN13 input　　後段 DSP との間にあるＤＰＲＡＭを使い、後段 DSP からデータを受け取るデバイスドライバブロックです。
- CN12 output　　前段 DSP との間にあるＤＰＲＡＭを使い、前段 DSP へデータを渡すデバイスドライバブロックです。
- CN13 output　　後段 DSP との間にあるＤＰＲＡＭを使い、後段 DSP へデータを渡すデバイスドライバブロックです。

＊前段・後段ＤＳＰについては、“ＤＳＰの前段と後段の意味”（91ページ）を参照してください。

2.11.2 コネクタとボードとのメモリマップ

ＡＤＳＰ３２４－１４３は、ＤＳＰボード１枚に対して最大４台使用できます。コネクタ番号とＤＳＰボードのメモリ対応は、表 2.11.1 の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

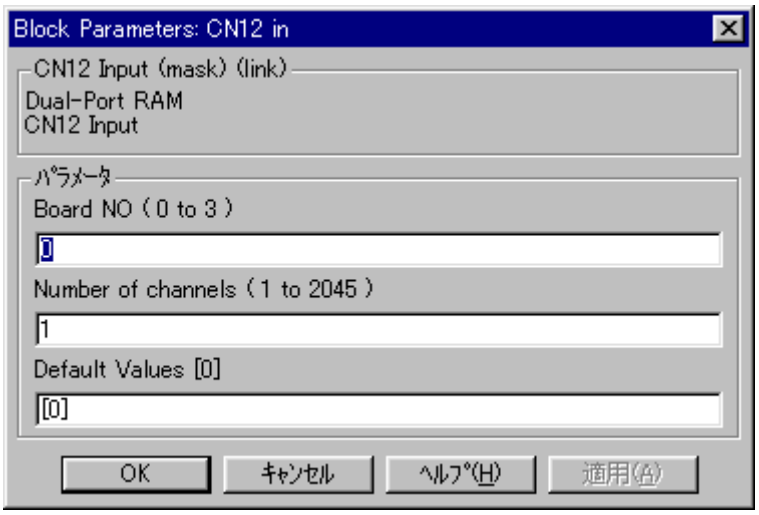
表 2.11.1　コネクタとＡＤＳＰ３２４－１４３ボードメモリマップ

ボード	ボード番号	ＣＮ１２	ＣＮ１３
１台目	0	908000ｈ～908FFFh	909000ｈ～909FFFｈ
２台目	1	90C000h～90CFFFｈ	90D000h～90DFFFｈ
３台目	2	910000ｈ～910FFFｈ	911000ｈ～911FFFｈ
４台目	3	914000ｈ～914FFFｈ	915000ｈ～915FFFｈ

2.11.3 ＣＮ１２　Ｉｎｐｕｔデバイスドライバブロック

ＣＮ１２　Ｉｎｐｕｔデバイスドライバーは、コネクタ１２を使用し、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いて、前段ＤＳＰとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ＤＳＰボード間の通信をホストコンピュータのバスを介さずに直接ＤＳＰボード間で交信することができます。

図 2.11-2　ADSP324-143 CN12　Input



- Board NO (0 to 3)

  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1を参照してください。

- Number of channels (1 to 2045)

  相手 DSP から受け取るデータのチャネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Values [0]

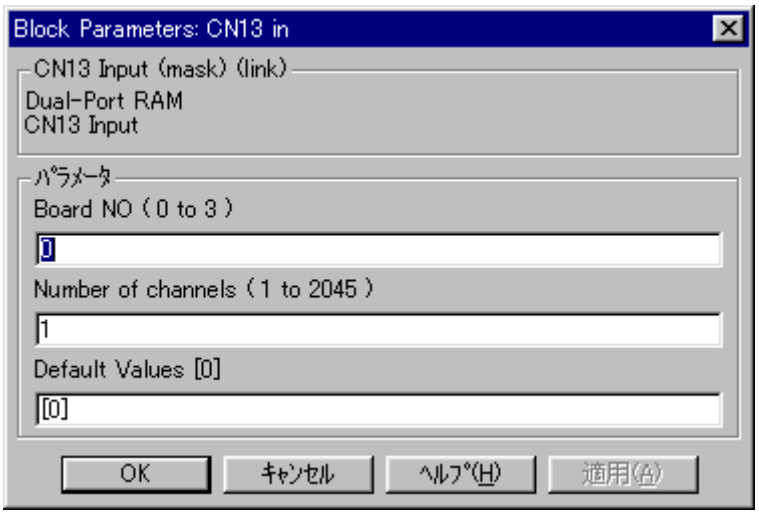
  相手 DSP が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル（[ ]）を与えます。空ベクトル時は、第二引数の幅のデータ０ベクトルを与えたことと同一になります。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“ＤＳＰ間ＤＰ通信アービトレーション”（90ページ）を参照してください。
又、前段・後段ＤＳＰについては、“ＤＳＰの前段と後段の意味”（91ページ）を参照してください。

2.11.4 ＣＮ１３　Ｉｎｐｕｔデバイスドライバブロック

ＣＮ１３　Ｉｎｐｕｔデバイスドライバーは、コネクタ１３を使用し、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いて、後段ＤＳＰとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ＤＳＰボード間の通信をホストコンピュータのバスを介さずに直接ＤＳＰボード間で交信することができます。

図 2.11-3　ADSP324-143 CN13　Input



- Board NO (0 to 3)

  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1を参照してください。

- Number of channels (1 to 2045)

  相手 DSP から受け取るデータのチャネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Values [0]

  相手 DSP が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル（[ ]）を与えます。空ベクトル時は、第二引数の幅のデータ０ベクトルを与えたことと同一になります。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、
“ＤＳＰ間ＤＰ通信アービトレーション”（90p）を参照してください。
又、前段・後段ＤＳＰについては、“ＤＳＰの前段と後段の意味”（91ｐ）を参照してください。

2.11.5 ＤＳＰ間ＤＰ通信アービトレーション

ＤＳＰ間のＤＰＲＡＭ通信デバイスブロックによる交信は、データの同時性（ブロックから取り出したデータ（送ったデータ）が全て同一時刻である事）を保証しています。つまり、「データの前半が後半より１時刻（ステップサイズ）分、昔のデータであった」等ということが無いように、送る側と受ける側でフラグを用いたソフトウェア同期処理を行っています。
これは、複数のＤＳＰが異なる周期で計算を行う、非同期処理間でデータを交換する為には必要な機構です。

本ソフトウェアでは、データの同時性は、以下の考え方によりアービトレーションしています。

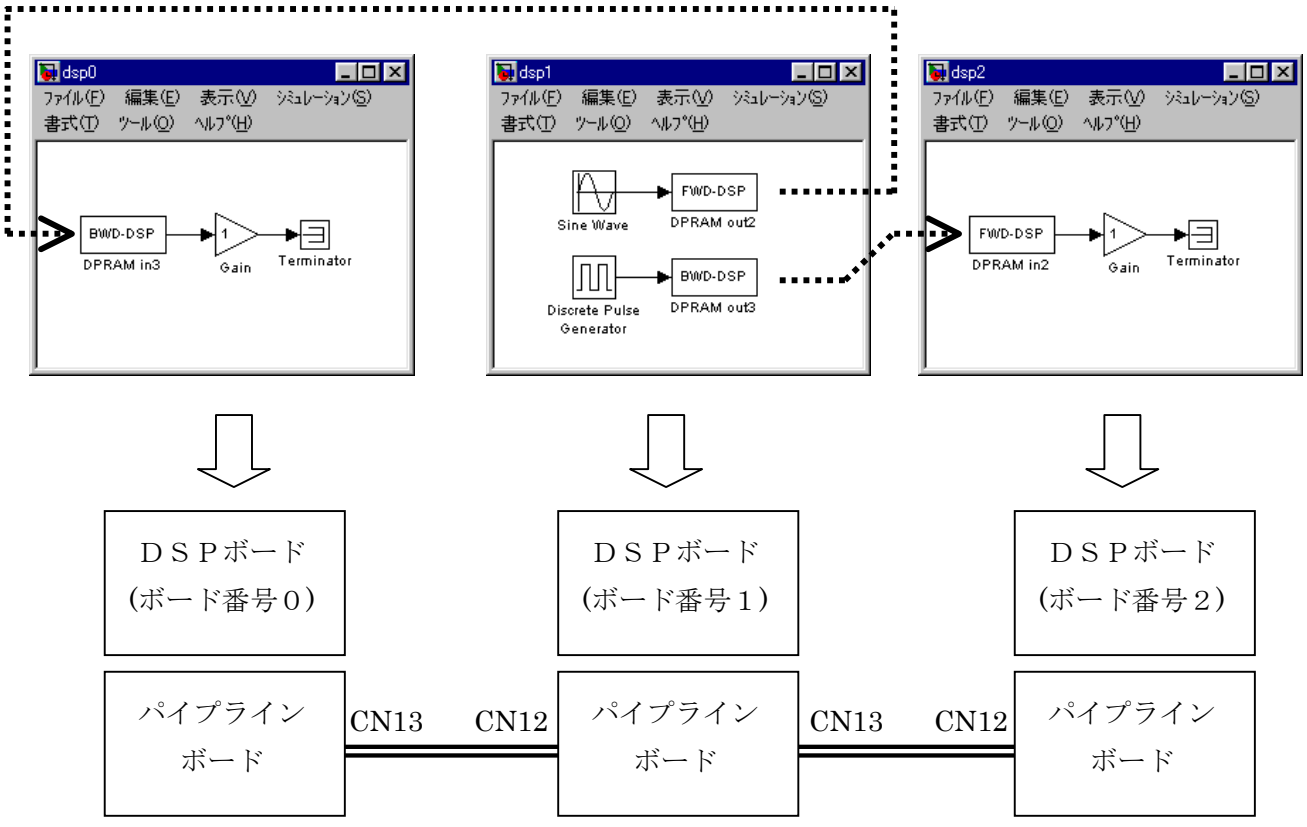
１、書き込み側の要求が先に発生した場合
　　読み出し側は書き込み側が完了するまで待機して後読み出す。
２、読み出し側の要求が先に発生した場合
　　書き込み側は読み出しみ側が完了するまで待機して後書き込む。
３、両方の要求が同時に発生した場合
　　読み出し側は書き込み側が完了するまで待機して後読み出す。

上記３項は、同時の場合は、最新のデータを渡す為、読み出し側が待機します。

2.11.6 ＤＳＰの前段と後段の意味

ＤＳＰ間のＤＰＲＡＭ通信デバイスブロックには、前段（ＦＷＤ）と後段（ＢＷＤ）があります。前段とは当該ＤＳＰボードに接続されているパイプラインボード（ADSP324-143等）のコネクターＣＮ１２の先に接続されたＤＳＰボードであり、後段とは同ＣＮ１３の先に接続されているＤＳＰボードの事を表します。つまり、前段・後段の関係は、パイプラインボードの接続がどうなっているかにより決定され、個々のＤＳＰボードのボード番号には全く関係ありません。

以下に簡単な例を示します。説明の便宜上各ＤＳＰにはボード番号０、１、２と番号を付けたとします。下の図では、真中のＤＳＰボード（ボード番号１）から見て前段とは、同ボードに接続されたパイプラインボードのＣＮ１２に接続されたＤＳＰボードですから、ボード番号０が前段になり、同様に後段は、同ボードに接続されたパイプラインボードのＣＮ１３に接続されたＤＳＰボードですから、ボード番号２が後段になります。 図のブロック線図を結ぶ破線矢印 ⇢ は信号の流れを表しています。

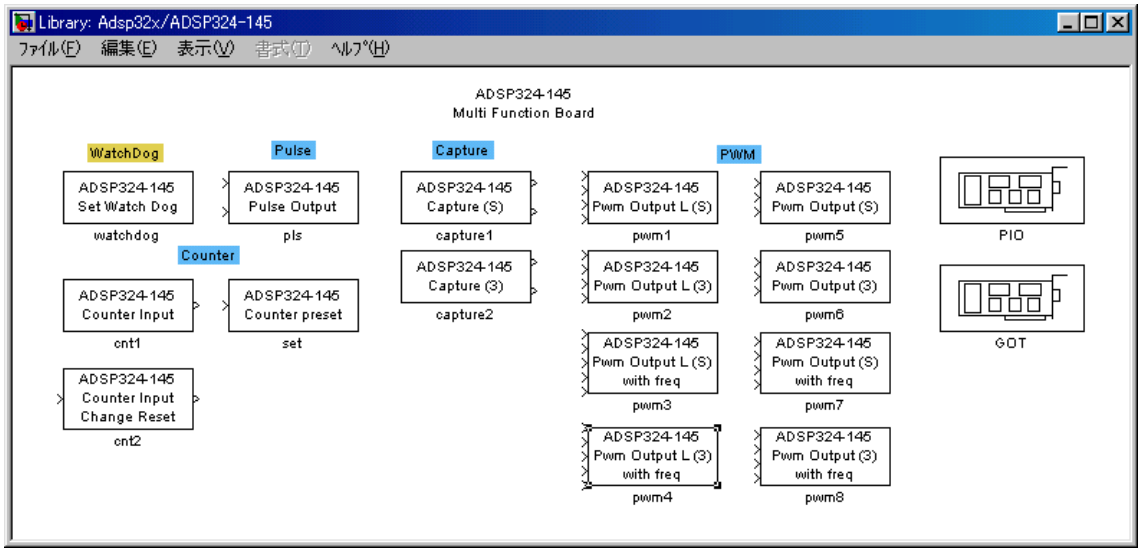


2.12ＡＤＳＰ３２４－１４５デバイスドライバーブロック

2.12.1 概要

図 2.12-1に、ＡＤＳＰ３２４－１４５用デバイスドライバウィンドウを開いた所を示します。

図 2.12-1　ADSP324-145 ドライバウィンドウ



この中にはＡＤＳＰ３２４－１４５とのインターフェースを取るための２２のデバイスドライバーブロックが登録されています。

- watch dog　　ウォッチドグタイマ
- Pulse Output　　パルス出力デバイスドライバです。
- Counter Input　　プリセット付カウンタデバイスドライバです。
- Counter preset　　カウンタープリセットドライバです。
- Capture[S]　　単相モードのキャプチャデバイスドライバです。
- Capture[3]　　３相モードのキャプチャデバイスドライバです。
- Pwm Output[S]　　単相モードのＰＷＭ出力デバイスドライバです。
- Pwm Output[3]　　３相モードのＰＷＭ出力デバイスドライバです。
- PIO Set Time　　PIO のﾊﾝﾄﾞﾘﾝｸﾞ信号の ON/遅延時間設定デバイスドライバです。
- PIO bit In[x]　　ビット入力デバイスドライバです。
- PIO Paralell In[x]　複数ビット同時入力デバイスドライバです。
- PIO bit Out[x]　　ビット出力デバイスドライバです。
- PIO Paralell Out[x]　複数ビット同時出力デバイスドライバです。
- GOT　　GOT 操作用デバイスドライバです。

2.12.2 コネクタとボードとのメモリマップ

ＡＤＳＰ３２４－１４５は、ＤＳＰボード１枚に対して最大４台使用できます。コネクタ番号とＤＳＰボードのメモリ対応は、表 2.12.1 の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表 2.12.1　コネクタとＡＤＳＰ３２４－１４５ボードメモリマップ

ボード	ボード番号	ADSP324-145	
		ADSP324-00A	ADSP674 ｼﾘｰｽﾞ
１台目	0	00901000h	03004000h
２台目	1	00901200h	03004800h
３台目	2	00901400h	03005000h
４台目	3	00901600h	03005800h

表 2.12.2　ＡＤＳＰ３２４-１４５カウンタチャネル番号対応

ボード	カウンタチャンネル番号				絶縁ポート番号
	ｶｳﾝﾀｰ０	ｶｳﾝﾀｰ１	ｶｳﾝﾀｰ２	ｶｳﾝﾀｰ３	
１台目	０	１	２	３	０
２台目	４	５	６	７	１
３台目	８	９	１０	１１	２
４台目	１２	１３	１４	１５	３

表 2.12.3　ＡＤＳＰ３２４-１４５キャプチャ・ＰＷＭ・パルスチャネル番号対応

ボード	キャプチャ・ＰＷＭ・パルスチャンネル番号			
	ＣＨ０	ＣＨ１	ＣＨ２	ＣＨ３
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

表 2.12.4　ＡＤＳＰ３２４-１４５ＰＩＯポート番号対応

ボード	ポート０ ＯＵＴ	ポート０ ＩＮ	ポート２ ＯＵＴ	ポート２ ＩＮ
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

表 2.12.5　ＡＤＳＰ３２４－１４５ボードPI0とビット番号の対応

入出力信号名	ビット番号 注１	パラレル 注２	ポート単位 注３
SIG031 / SIG331	31	最後	2^{31}
SIG030 / SIG330	30	最後の１つ前	2^{30}
・	・	・	・
・	・	・	・
SIG006 / SIG306	6	・	2^{6}
SIG005 / SIG305	5	・	2^{5}
SIG004 / SIG304	4	・	2^{4}
SIG003 / SIG303	3	・	2^{3}
SIG002 / SIG302	2	・	2^{2}
SIG001 / SIG301	1	先頭の次	2^{1}
SIG000 / SIG300	0	先頭	2^{0}

注１：ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

注２：パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注３：ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

表 2.12.6　ＰＩＯハンドリング方式

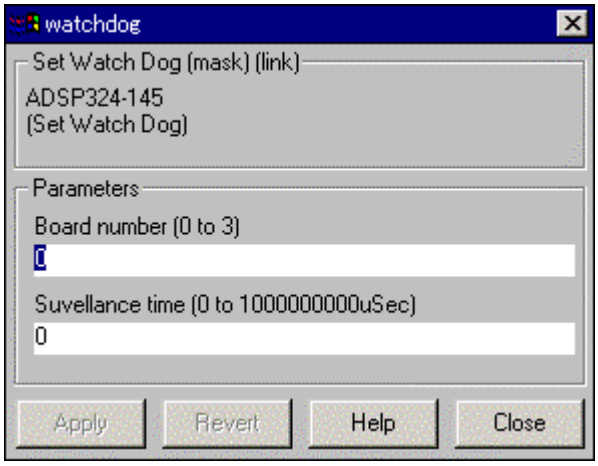
	ハンドリング方式
0	ハンドリングなし
1	ホールド方式
2	Strobe / Ack 方式
3	Write Strobe 方式

注１：ハンドリング方式の詳細は、“ＡＤＳＰ３２４－１４５ハードウェア・ユーザーズ・マニュアル”機能説明（パラレル入出力　ハンドリング方式）を参照してください。

2.12.3 WatchDog デバイスドライバブロック

WatchDog デバイスドライバーは、ウォッチドグタイマを使用可能にします。

図 2.12-2　ADSP324-145 WatchDog



- Board　NO (0 to 3)

  ADSP324-145 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表 2.12.1を参照してください。

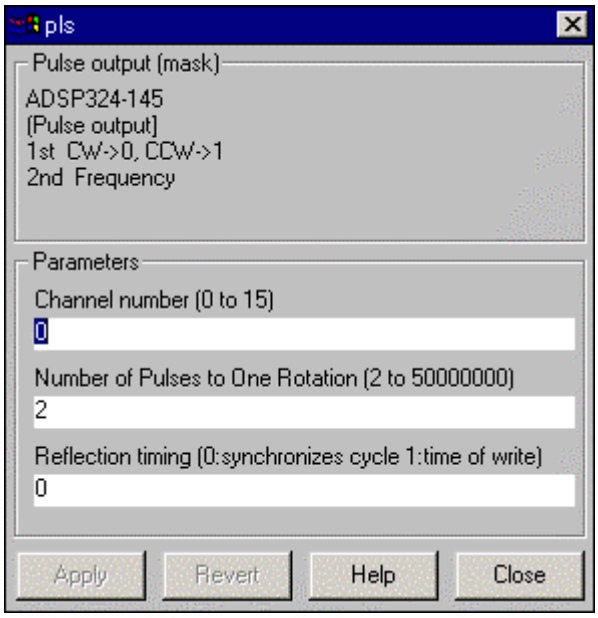
- Surveillance　time〔0 to 1000000mSec〕

  ウォッチドグタイマの監視周期時間を 0～1000000 mSec で設定してください。

2.12.4　Pulse output デバイスドライバブロック

Pulse output　デバイスドライバーは、第一入力として、回転方向（０：正転＜ＣＷ＞、１逆転＜ＣＣＷ＞)、第二入力としてパルス周波数（０．０１Ｈｚ～６ＭＨｚ）を入力すると、指定チャネルへパルス出力します。

図 2.12-3　ADSP324-145 Pulse output



- Channel number (0 to 15)

  ADSP324-145 のパルスチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

- Number of Pulses to One Rotation 〔1 to 50000000〕

  １回転のパルス数を指定します。パルス数とは、Z 相の出力をＡ・B 相パルスの何パルスに１回出力するかを設定します。

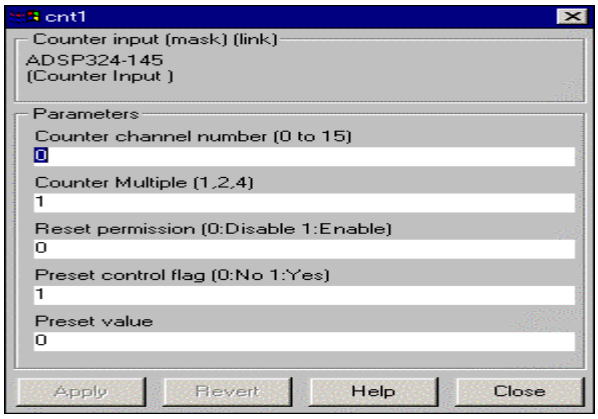
- Reflect Timing〔0: frequency 1: immediately〕

  パルス設定値の反映タイミングを０：周波数に同期、１：書込み時のいずれかを指定します。

2.12.5 Counter Input デバイスドライバブロック

Counter Input デバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンターは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。又リセットを有効にした場合リセット条件が満たされた時カウンタがリセットされます。

図 2.12-4　ADSP324-145 Counter Input



- Counter channel number (0 to 15)

  入力の対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表 2.12.2を参照してください。

- Counter Multiple (1,2,4)

  カウンタの逓倍を　１：×１、２：×２、４：×４　のいずれかより選択してください。

- Reset permission (0:Disable 1: Enable)

  カウンタリセット信号の許可を設定してください。（０：未許可、１：許可）
  詳細は、ADSP324-145 ハードウェアマニュアルを参照してください。

- Preset controll flag (0:No 1:Yes)

  初期化時にカウンタ値のプリセットを行なうか否かを決定します。０又は１を設定してください。０を設定した場合はプリセットは行ないません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。

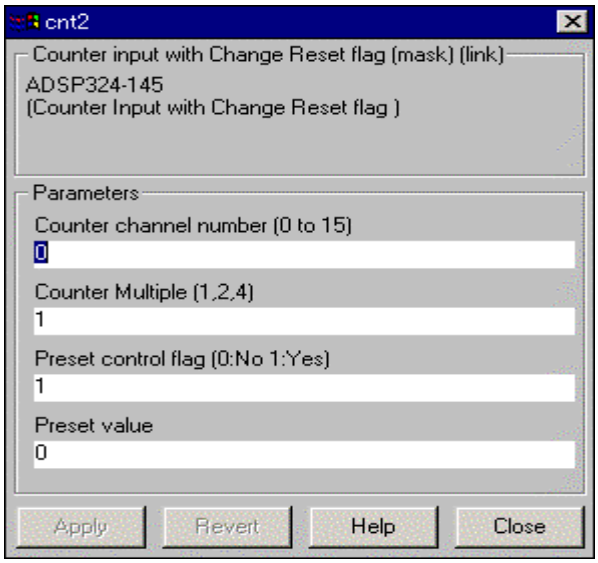
- Preset value

  Preset controll flag に１を設定した場合、ここに設定した値がカウンターの初期値として採用されます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

2.12.6 Counter lnput with Change reset　flag デバイスドライバブロック

Counter lnput with Change Reset flag デバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。リセット許可（０：未許可　１：許可）をスカラーで入力します。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンターは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。

図 2.12-5　ADSP324-145 Counter lnput with Change Reset flag



- Counter channel number (0 to 15)

  入力の対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表 2.12.2を参照してください。

- Counter Multiple (1,2,4)

  カウンタの逓倍を　１：×１、２：×２、４：×４　のいずれかより選択してください。

- Preset controll flag (0:No 1:Yes)

  初期化時にカウンタ値のプリセットを行なうか否かを決定します。０又は１を設定してください。０を設定した場合はプリセットは行ないません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。

- Preset value

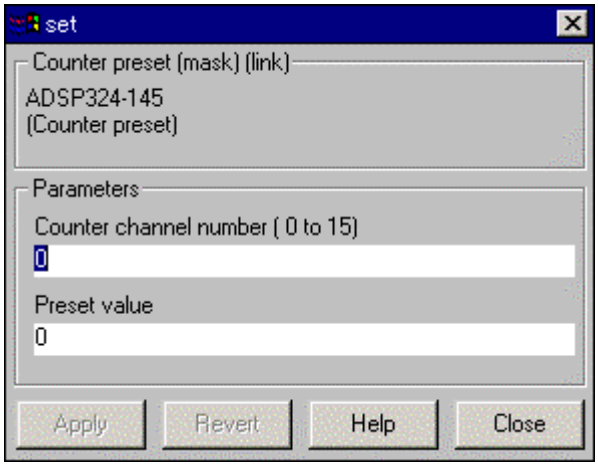
  Preset controll flag に１を設定した場合、ここに設定した値がカウンターの初期値として採用されます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

2.12.7 Counter preset デバイスドライバブロック

Counter preset デバイスドライバブロックは、任意のチャンネル番号のカウンターを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に１度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。

このブロックには１つのスカラー入力があり、ここに０又は１の値を入力することによりプリセットタイミングを制御します。０が入力されている期間はカウンタはカウント動作を行ないます。１が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図 2.12-6　ADSP324-145 Counter preset



- Counter channel number (0 to 15)

  対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表 2.12.2を参照してください。

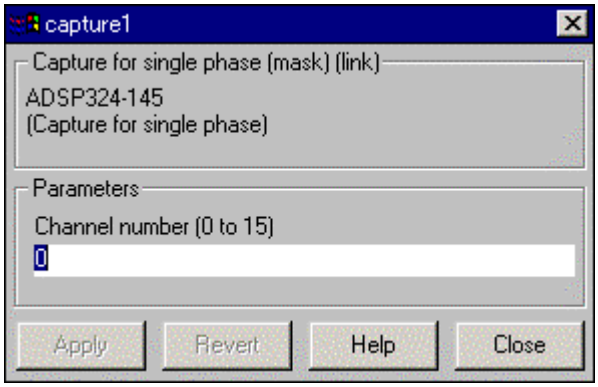
- Preset value

  ここに設定した値がカウンターにセットされます。設定できる値は、符号付３２ビット整数（ストレートバイナリ）で表せる範囲（－2^{31}～2^{31}－１）です。

2.12.8 Capture(S) デバイスドライバブロック

Capture(S)デバイスドライバーは、指定チャネルの単相信号を計測し、第一出力に周期（周波数)、第二出力にＵＶ　ＯＮ時間（Duty 比 100.0～-100.0）を出力します。

図 2.12-7　ADSP324-145 Capture〔S〕



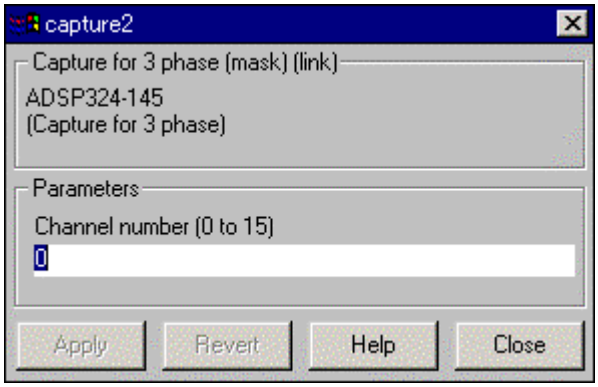
- Channel number (0 to 15)

  ADSP324-145 のキャプチャチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

2.12.9　Capture(3)デバイスドライバブロック

Capture(3) デバイスドライバーは、指定チャネルの単相信号を計測し、第一出力に周期（周波数)、第二出力にＯＮ時間（Duty 比 100.0～-100.0）をＵＶ相・ＶＷ相・ＷＵ相の順にベクトル出力します。

図 2.12-8　ADSP324-145 Capture(3)



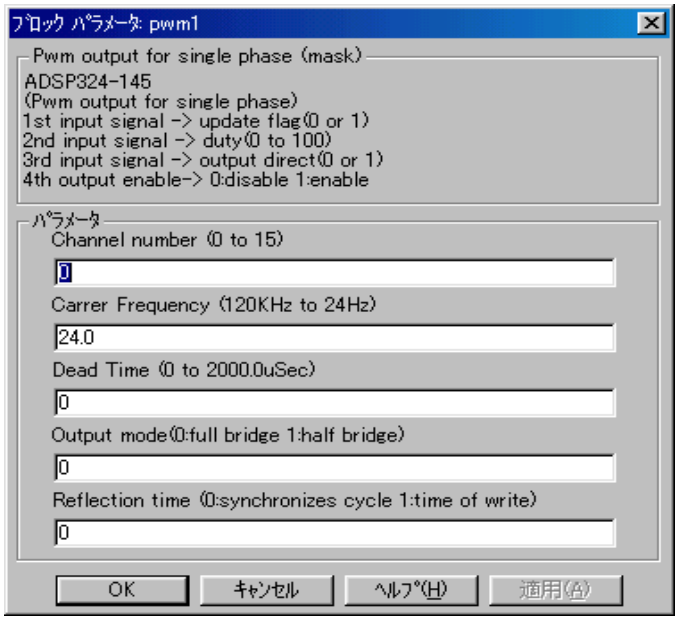
- Channel number (0 to 15)

  ADSP324-145 のキャプチャチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

2.12.10　Pwm Output L〔S〕デバイスドライバブロック

Pwm Output〔S〕デバイスドライバーは、第一入力としてＤｕｔｙ比の変更有無（１：変更する、０：変更しない）第二入力としてＤｕｔｙ比（0.0～100.0%)、第三入力に出力方向（０：Ｐ側出力、１：Ｎ側出力）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第四入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 2.12-9　ADSP324-145 Pwm Output(S)



- Channel number (0 to 15)

  ADSP324-145 のＰＷＭチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

- Carrer Frequency 〔120kHz to 24Hz〕

  キャリア周波数を１２０ＫＨｚから２４Ｈｚにて指定してください。

- Dead Time〔0 to 2000.0 μSec〕

  デットタイムを０～２０００．０μＳｅｃにて指定してください。

- Output Mode〔0:full bridge　1:half bridge〕

  波形の出力モードを指定してください。

  ０：フルブリッジ出力　　１：ハーフブリッジ出力

- Reflection time〔0:synchronizes cycle 1:time of write〕

  設定値の反映タイミングを指定してください。

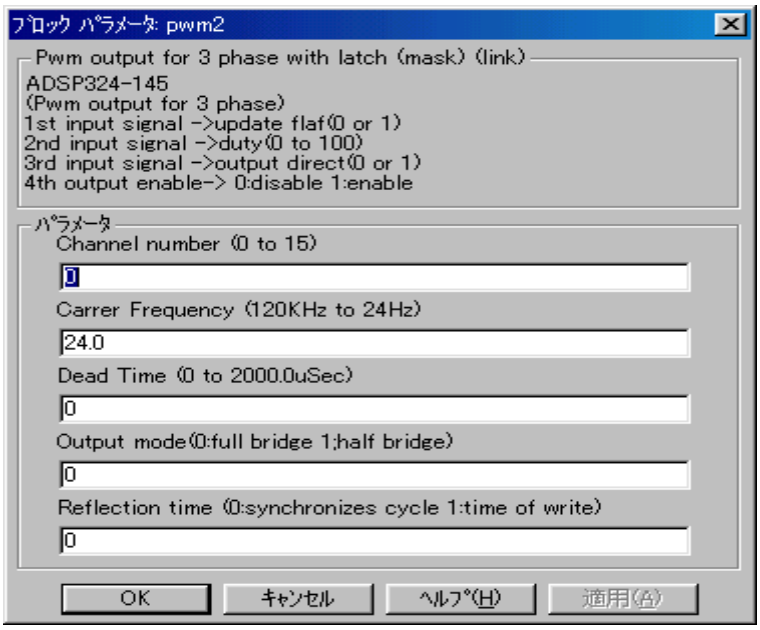
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）

  １：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.11　Pwm Output L (3)デバイスドライバブロック

Pwm Output(3)　デバイスドライバーは、第一入力としてＤｕｔｙ比の変更有無（１：変更する、０：変更しない）第二入力としてＤｕｔｙ比（0.0～100.0%）をU相・V相・W相の順にベクトル入力、第三入力に出力方向（０：Ｐ側出力、１：Ｎ側出力）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第四入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 2.12-10　ADSP324-145 Pwm Output(3)



- Channel number (0 to 15)

  ADSP324-145 のＰＷＭチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

- Carrer Frequency 〔120kHz to 24Hz〕

  キャリア周波数を１２０ＫＨｚから２４Ｈｚにて指定してください。

- Dead Time〔0 to 2000.0 μSec〕

  デットタイムを０～２０００．０μＳｅｃにて指定してください。

- Output Mode〔0:full bridge　1:half bridge〕

  波形の出力モードを指定してください。

  ０：フルブリッジ出力　　１：ハーフブリッジ出力

- Reflection time〔0:synchronizes cycle 1:time of write〕

  設定値の反映タイミングを指定してください。

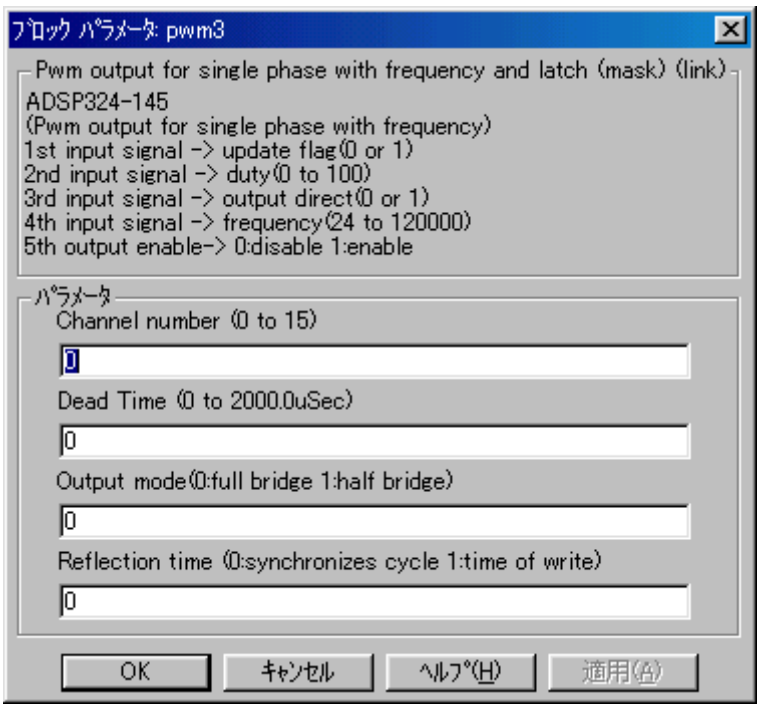
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）

  １：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.12　Pwm Output L（S）with Freq デバイスドライバブロック

Pwm Output (S) with Freq デバイスドライバーは、第一入力としてＤｕｔｙ比の変更有無（１：変更する、０：変更しない）第二入力としてＤｕｔｙ比（0.0～100.0%)、第三入力に出力方向（０：Ｐ側出力、１：Ｎ側出力)、第四入力としてキャリア周波数（120KHz～24Hz）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第五入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 2.12-11　ADSP324-145 Pwm Output (S) with Freq


ﾌﾞﾛｯｸ ﾊﾟﾗﾒｰﾀ: pwm3
Pwm output for single phase with frequency and latch (mask) (link)
ADSP324-145
(Pwm output for single phase with frequency)
1st input signal -> update flag(0 or 1)
2nd input signal -> duty(0 to 100)
3rd input signal -> output direct(0 or 1)
4th input signal -> frequency(24 to 120000)
5th output enable-> 0:disable 1:enable
ﾊﾟﾗﾒｰﾀ
Channel number (0 to 15)
0
Dead Time (0 to 2000.0uSec)
0
Output mode(0:full bridge 1:half bridge)
0
Reflection time (0:synchronizes cycle 1:time of write)
0
OK
ｷｬﾝｾﾙ
ﾍﾙﾌﾟ(H)
適用(A)

- Channel number (0 to 15)

  ADSP324-145 のＰWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

- Dead Time〔0 to 2000.0 μSec〕

  デットタイムを０～２０００．０μＳｅｃにて指定してください。

- Output Mode〔0:full bridge　1:half bridge〕

  波形の出力モードを指定してください。

  ０：フルブリッジ出力　　１：ハーフブリッジ出力

- Reflection time〔0:synchronizes cycle 1:time of write〕

  設定値の反映タイミングを指定してください。

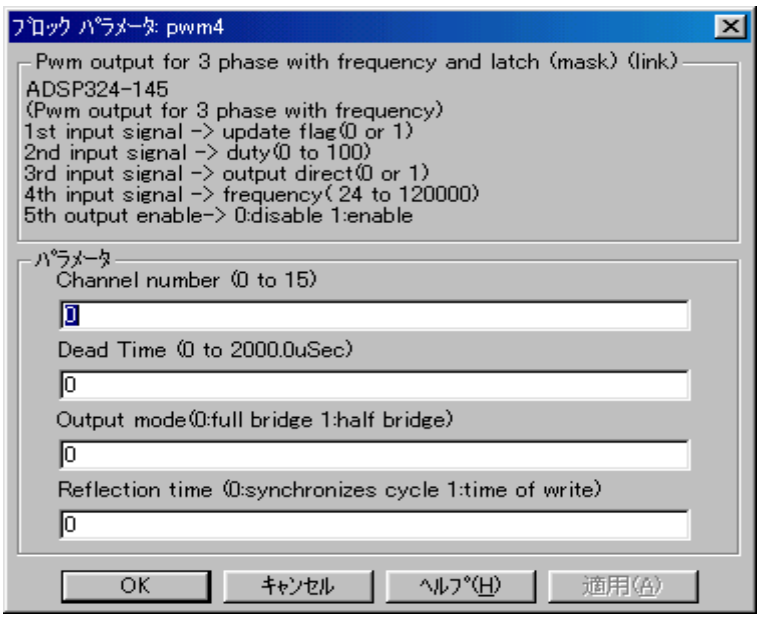
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）

  １：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.13　　Pwm Output L (3) with Freq デバイスドライバブロック

Pwm Output (3) with Freq デバイスドライバーは、第一入力としてＤｕｔｙ比の変更有無（１：変更する、０：変更しない)、第二入力としてＤｕｔｙ比（0.0～100.0%）をU相・Ｖ相・W相の順にベクトル入力、第三入力に出力方向（０：Ｐ側出力、１：Ｎ側出力)、第四入力としてキャリア周波数（120KHz～24Hz）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第五入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 2.12-12　ADSP324-145 Pwm Output (3) with Freq


ブロック パラメータ: pwm4
Pwm output for 3 phase with frequency and latch (mask) (link)
ADSP324-145
(Pwm output for 3 phase with frequency)
1st input signal -> update flag(0 or 1)
2nd input signal -> duty(0 to 100)
3rd input signal -> output direct(0 or 1)
4th input signal -> frequency(24 to 120000)
5th output enable-> 0:disable 1:enable
パラメータ
Channel number (0 to 15)
0
Dead Time (0 to 2000.0uSec)
0
Output mode(0:full bridge 1:half bridge)
0
Reflection time (0:synchronizes cycle 1:time of write)
0
OK
キャンセル
ヘルプ(H)
適用(A)

- Channel number (0 to 15)

  ADSP324-145 のＰWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 2.12.3を参照してください。

- Dead Time〔0 to 2000.0 μSec〕

  デットタイムを０～２０００．０μＳｅｃにて指定してください。

- Output Mode〔0:full bridge　1:half bridge〕

  波形の出力モードを指定してください。

  ０：フルブリッジ出力　　１：ハーフブリッジ出力

- Reflection time〔0:synchronizes cycle 1:time of write〕

  設定値の反映タイミングを指定してください。

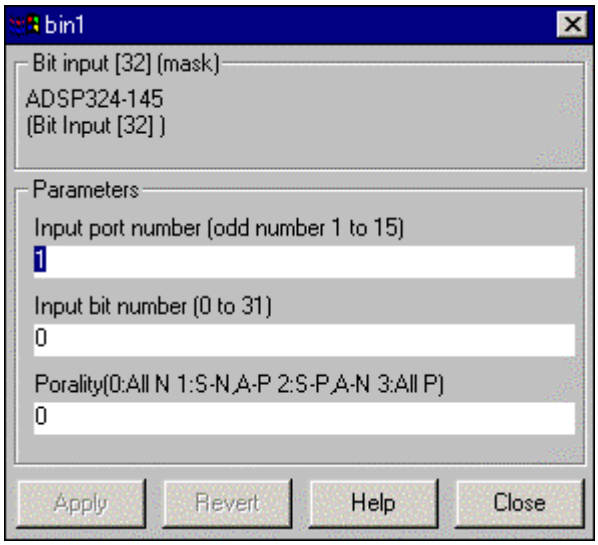
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）

  １：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.14　　PIO bit In〔32〕デバイスドライバブロック

PIO bit In〔32〕デバイスドライバーは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは３２ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時ＬＯＷ：１・ＨＩＧＨ：0 ・未接続：1,正論理選択時ＬＯＷ：0 ・ＨＩＧＨ：1 ・未接続：１です。ボードが接続されていないポートの全てのビットは不定です。

図 2.12-13　ADSP324-145　bit In〔32〕



- Input port number (odd number in 1 to 15)

  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。

- Input bit number (0 to 31)

  入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 2.12.5を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

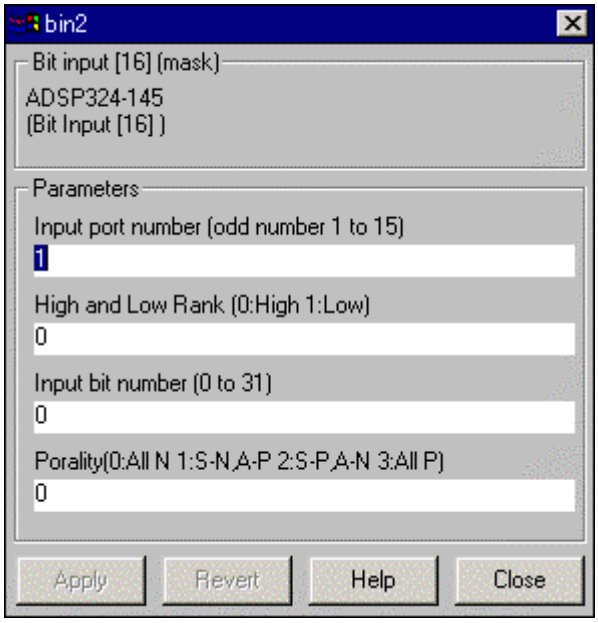
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.15　PIO bit In〔16〕デバイスドライバブロック

PIO bit In〔16〕デバイスドライバーは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは１６ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時ＬＯＷ：１・ＨＩＧＨ：0 ・未接続：1,正論理選択時ＬＯＷ：0 ・ＨＩＧＨ：1 ・未接続：１です。ボードが接続されていないポートの全てのビットは不定です。

図 2.12-14　ADSP324-145　bit In〔16〕



- Input port number (odd number in 1 to 15)

  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Input bit number(0 to 15)

  入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 2.12.5を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

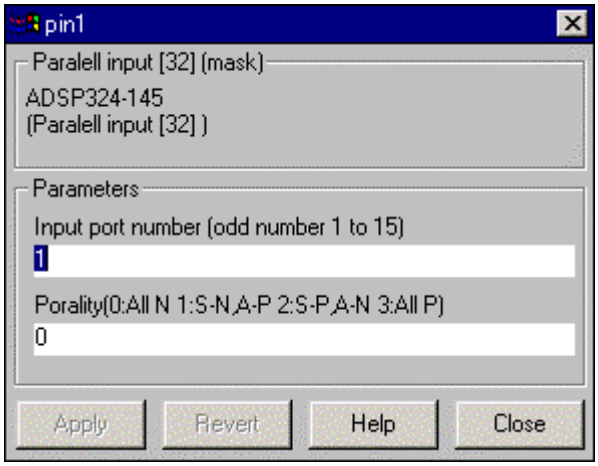
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.16　PIO paralell in〔32〕デバイスドライバブロック

PIO paralell in〔32〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの全ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-15　ADSP324-145　paralell in〔32〕



- Input port number (odd number in 1 to 15)

  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

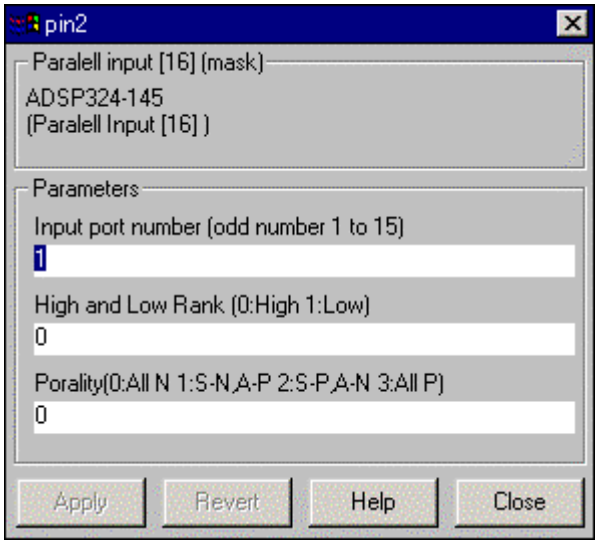
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.17　PIO paralell in〔16〕デバイスドライバブロック

PIO paralell in〔16〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは、幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-16　ADSP324-145　paralell in〔16〕



- Input port number (odd number in 1 to 15)

  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

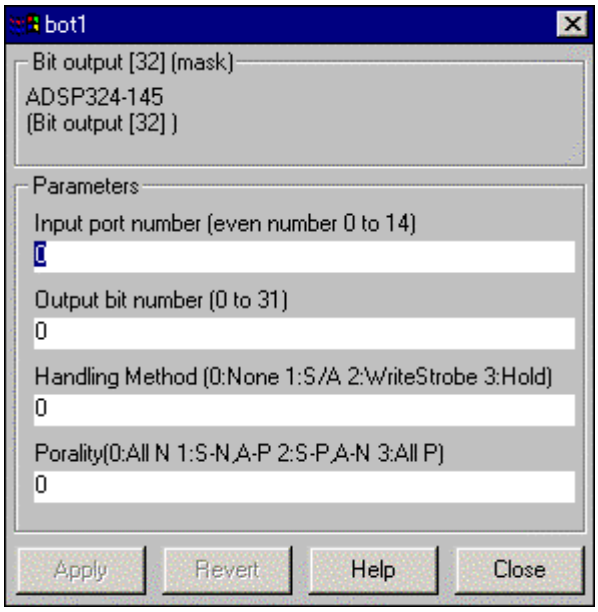
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.18　PIO bit out〔32〕デバイスドライバブロック

PIO bit out〔32〕デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは３２ビット単位のアクセスになります。ポードから出力される信号は、このブロックへの入力レベルが０の時ＯＦＦ、１の時ＯＮです。ボードの出力初期値は全ビット共ＯＦＦです。

図 2.12-17　ADSP324-145　bit out〔32〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Output bit number〔0 to 31〕

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

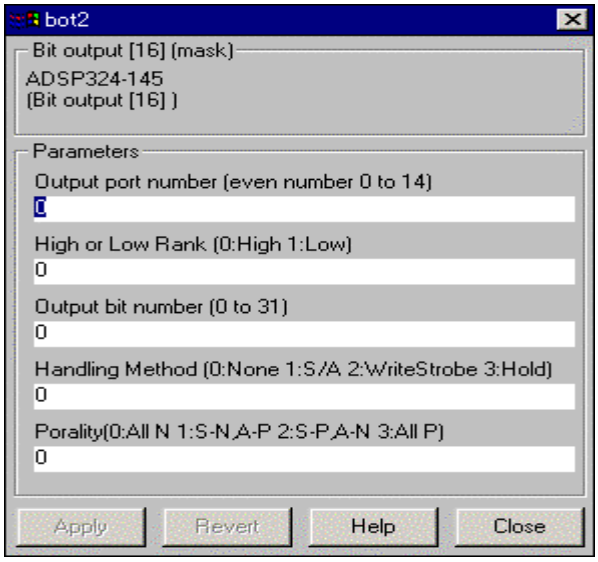
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.19　PIO bit out〔16〕デバイスドライバブロック

PIO bit out〔16〕デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは１６ビット単位のアクセスになります。ポードから出力される信号は、このブロックへの入力レベルが０の時ＯＦＦ、１の時ＯＮです。ボードの出力初期値は全ビット共ＯＦＦです。

図 2.12-18　ADSP324-145　bit out〔16〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Output bit number〔0 to 15〕

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

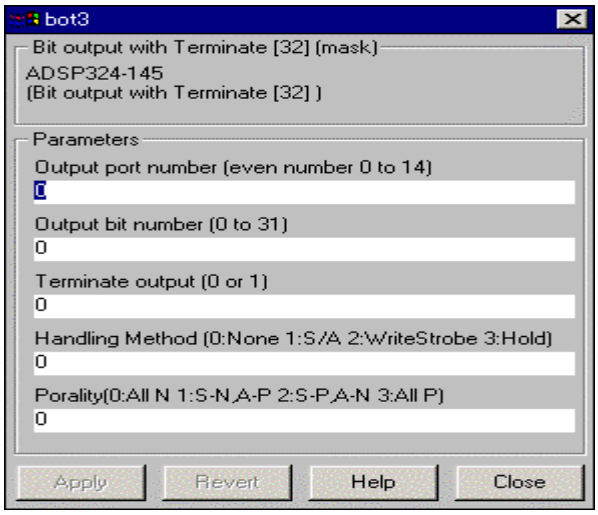
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.20　PIO bit out with Terminate〔32〕デバイスドライバブロック

PIO bit output Terminate out〔32〕デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは３２ビット単位のアクセスになります。ポードから出力される信号は、このブロックへの入力レベルが０の時ＯＦＦ、１の時ＯＮです。ボードの出力初期値は全ビット共ＯＦＦです。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-19　ADSP324-145　bit output Terminate out〔32〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Output bit number〔0 to 31〕

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。

- Treminate output〔0 or 1〕

  実時間モデルが終了時する際に出力したい値を設定します。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

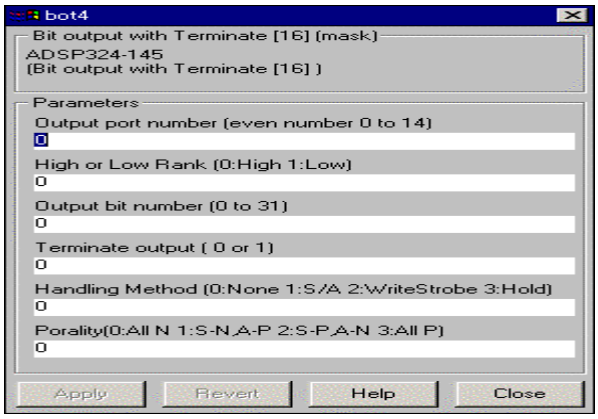
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.21　PIO bit out with Terminate〔16〕デバイスドライバブロック

PIO bit output Terminate out〔16〕デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは１６ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが０の時ＯＦＦ、１の時ＯＮです。ボードの出力初期値は全ビット共ＯＦＦです。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-20　ADSP324-145　bit output Terminate out〔16〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Output bit number〔0 to 31〕

  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。

- Treminate output〔0 or 1〕

  実時間モデルが終了時する際に出力したい値を設定します。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

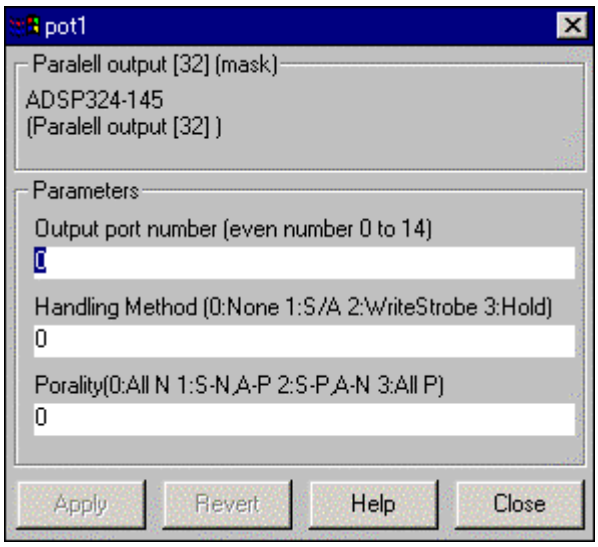
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.22　PIO paralell out〔32〕デバイスドライバブロック

PIO paralell out〔32〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの３２ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-21　ADSP324-145　paralell out〔32〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

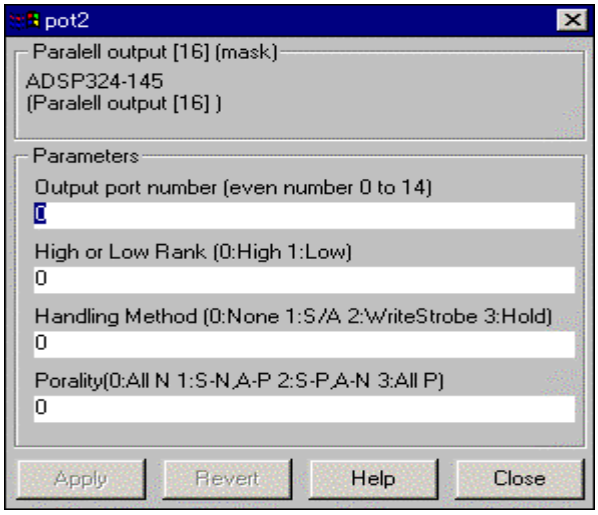
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.23　PIO paralell out〔16〕デバイスドライバブロック

PIO paralell out〔16〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは、幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-22　ADSP324-145　paralell out〔16〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

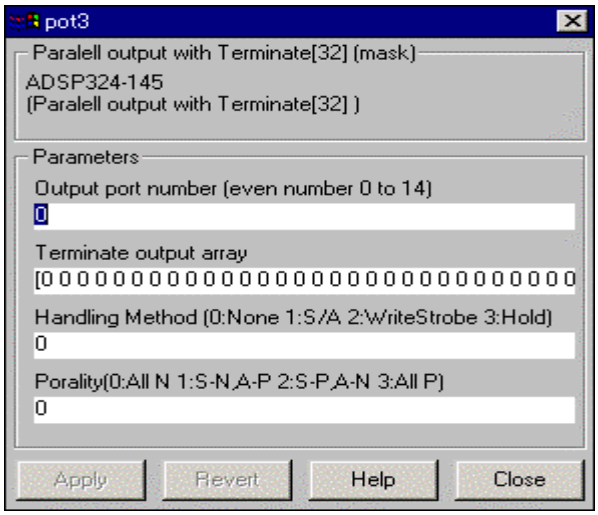
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.24　PIO parallel out with Terminate〔32〕デバイスドライバブロック

PIO paralell Terminate out〔32〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの３２ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-23　ADSP324-145　paralell Terminate out〔32〕


pot3
Paralell output with Terminate[32] (mask)
ADSP324-145
(Paralell output with Terminate[32])
Parameters
Output port number (even number 0 to 14)
0
Terminate output array
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Handling Method (0:None 1:S/A 2:WriteStrobe 3:Hold)
0
Porality(0:All N 1:S-N,A-P 2:S-P,A-N 3:All P)
0
Apply
Revert
Help
Close

- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Treminate output〔0 or 1〕

  実時間モデルが終了時する際に出力したい値を設定します。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

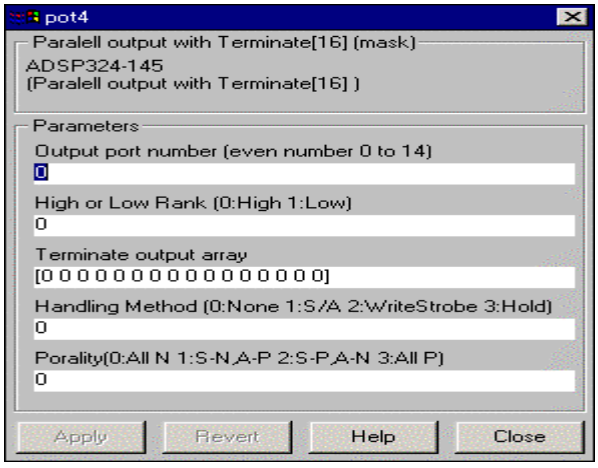
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.25　PIO paralell out with Terminate〔16〕デバイスドライバブロック

PIO paralell Terminate out〔16〕デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは、幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-24　ADSP324-145　paralell Terminate out〔16〕



- Output port number (even number in 0 to 14)

  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。

- Higher and Low Rank(0: Higer 1: Low)

  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Treminate output〔0 or 1〕

  実時間モデルが終了時する際に出力したい値を設定します。

- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕

  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

  入力信号の極性を指定します。

  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）

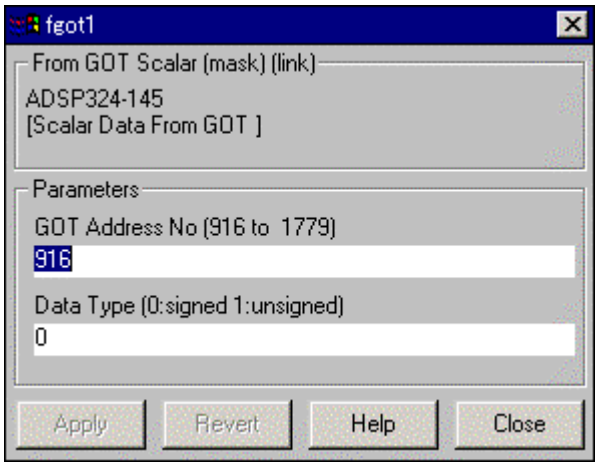
  ０：STB,ACK 共負論理　　１：STB 負論理、ACK 正論理

  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

2.12.26　From GOT デバイスドライバブロック

From GOT デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図 2.12-25　ADSP324-145　From GOT



- GOT Address No (916 to 1779)

  GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については “ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド” を参照してください。

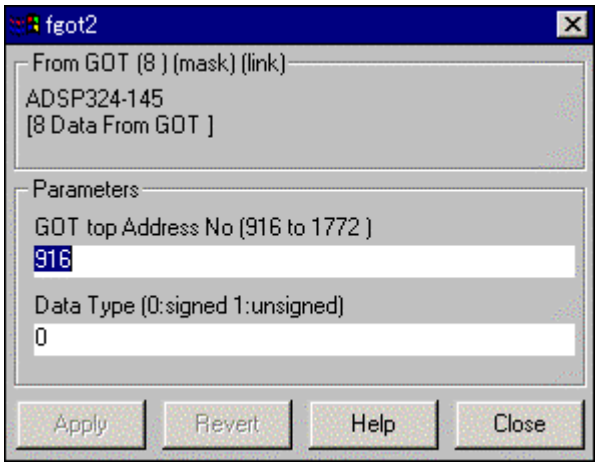
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.27　From GOT[8]デバイスドライバブロック

From GOT[8]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で取得します。

図 2.12-26　ADSP324-145　From GOT[8]



- GOT Address No (916 to 1772)

  GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

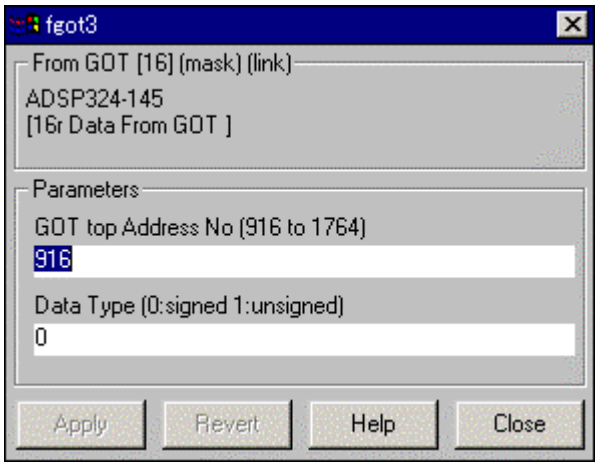
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.28　From GOT[16]デバイスドライバブロック

From GOT[16]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で取得します。

図 2.12-27　ADSP324-145　From GOT[16]



- GOT Address No (916 to 1764)

  GOT の先頭アドレス番号を 916～1764 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

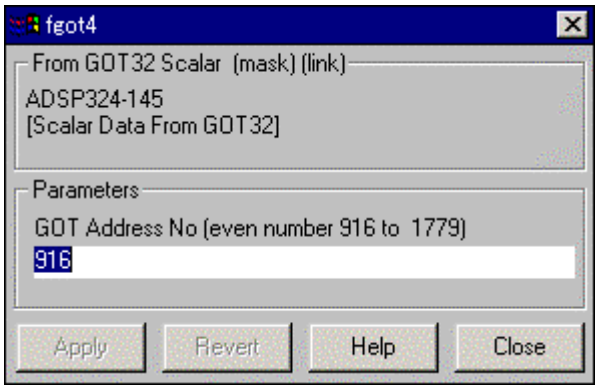
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.29　From GOT32 デバイスドライバブロック

From GOT32 デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図 2.12-28　ADSP324-145　From GOT32



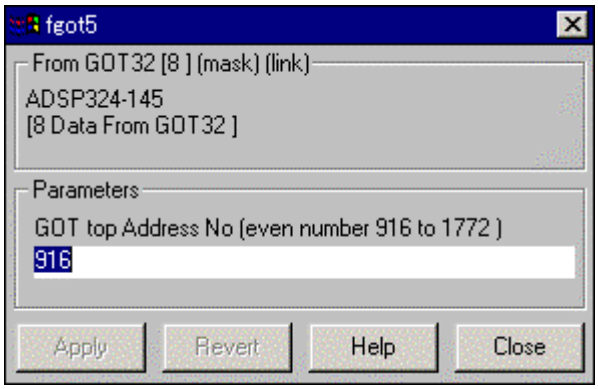
- GOT Address No (916 to 1779)

  GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については “ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.12.30　From GOT32 [8]デバイスドライバブロック

From GOT32 [8]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で取得します。

図 2.12-29　ADSP324-145　From GOT32 [8]



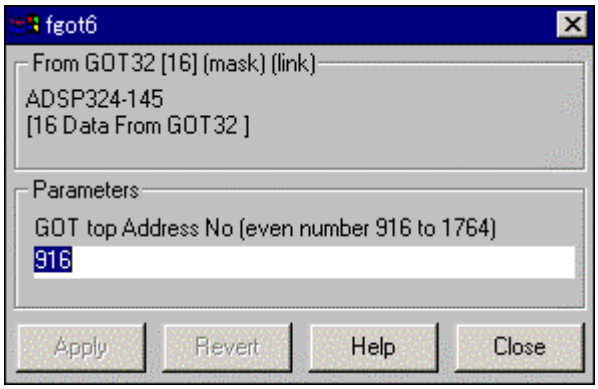
- GOT Address No (916 to 1772)

  GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.12.31　From GOT32 [16]デバイスドライバブロック

From GOT32 [16]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で取得します。

図 2.12-30　ADSP324-145　From GOT32 [16]



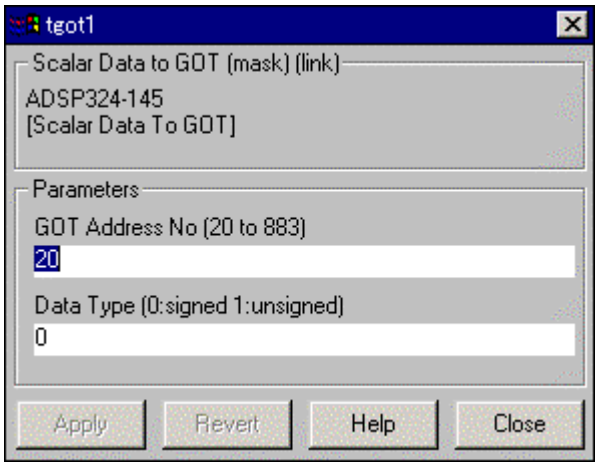
- GOT Address No (916 to 1764)

  GOT の先頭アドレス番号を 916～1764 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.12.32　To GOT デバイスドライバブロック

To GOT デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)　の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 2.12-31　ADSP324-145　To GOT



- GOT Address No (20 to 883)

  GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

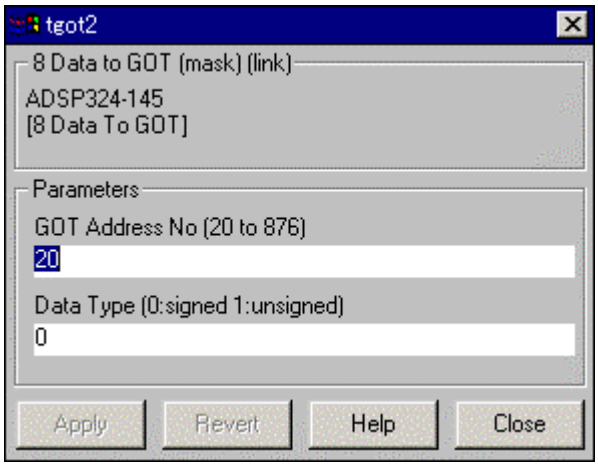
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.33　To GOT[8]デバイスドライバブロック

To GOT[8]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で出力します。

図 2.12-32　ADSP324-145　To GOT[8]



- GOT Address No (20 to 876)

  GOTの先頭アドレス番号を20～876の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

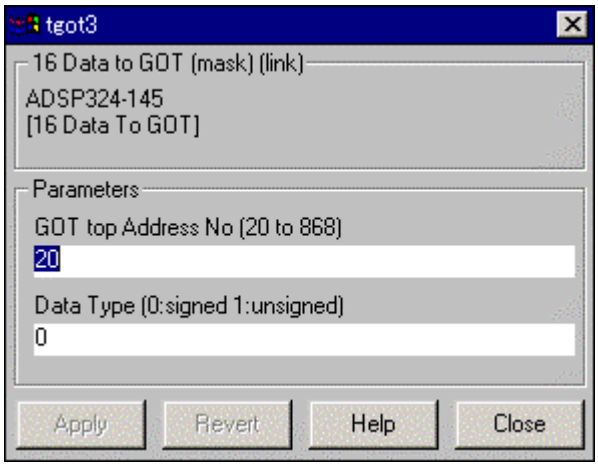
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.34　To GOT[16]デバイスドライバブロック

To GOT[16]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で出力します。

図 2.12-33　ADSP324-145　To GOT[16]



- GOT Address No (20 to 883)

  GOTの先頭アドレス番号を20～883の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

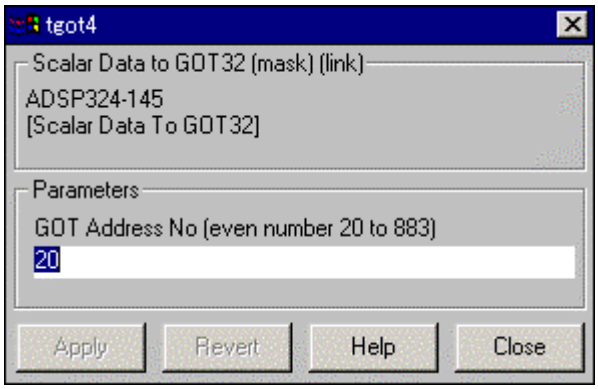
- Data Type (0:signed 1:unsigned)

  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ＤＳＰ操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.35　To GOT32 デバイスドライバブロック

To GOT32 デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ) の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 2.12-34　ADSP324-145　To GOT32



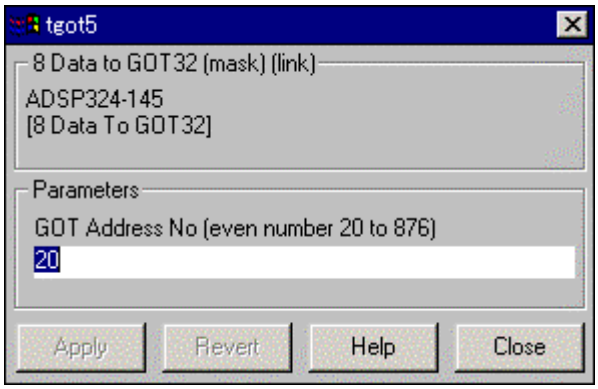
- GOT Address No (20 to 883)

  GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.12.36　To GOT32 [8]デバイスドライバブロック

To GOT32 [8]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で出力します。

図 2.12-35　ADSP324-145　To GOT32 [8]



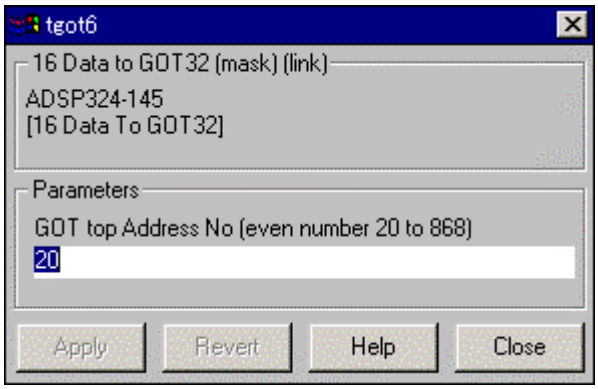
- GOT Address No (20 to 876)

  GOTの先頭アドレス番号を20～876の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.12.37　To GOT32 [16]デバイスドライバブロック

To GOT32 [16]デバイスドライバーは、ＤＳＰ操作ターミナル(ADSP-GOT シリーズ)の付属ＧＯＴ用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で出力します。

図 2.12-36　ADSP324-145　To GOT32 [16]



- GOT Address No (20 to 868)

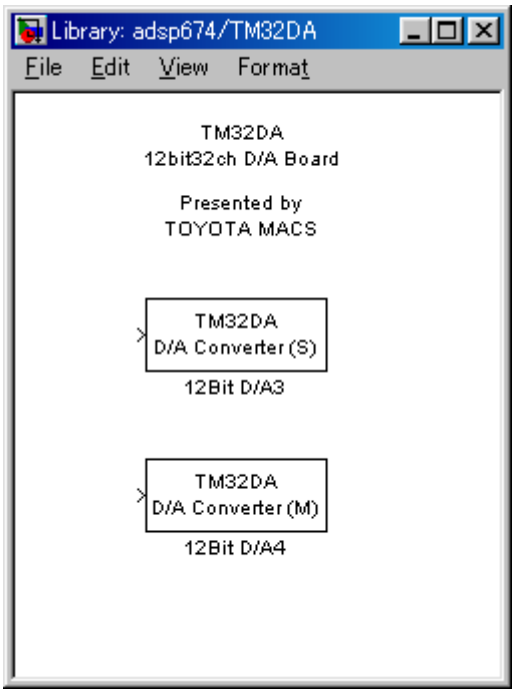
  GOTの先頭アドレス番号を20～868の範囲で指定してください。詳細については“ＤＳＰ操作ターミナル・ＧＯＴ操作ガイド”を参照してください。

2.13ＴＭ３２ＤＡデバイスドライバーブロック

2.13.1 概要

図 2.13-1に、ＴＭ３２ＤＡ（１２ビット多チャンネルＤ／Ａボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.13-1　TM32DA ドライバウィンドウ



この中にはＡＤＳＰ３２Ｘ－０３／５３とのインターフェースを取るための４つのデバイスドライバーブロックが登録されています。

- D/A input (S)　　１チャンネルのＤ／Ａ出力デバイスドライバブロックです。
- D/A input (M)　　多チャンネルのＤ／Ａ出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのＡ／Ｄ入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。Ｄ／Ａの場合も同様です。特に、Ａ／Ｄでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

表 2.13.1　チャンネル番号とＴＭ３２ＤＡボードとの対応

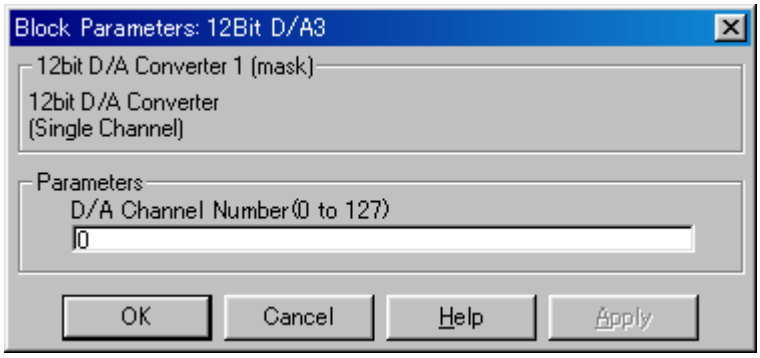
ボード	CH00	CH01	…	CH30	CH31
１台目	０	１	…	３０	３１
２台目	３２	３３	…	６２	６３
３台目	６４	６５	…	９４	９５
４台目	９６	９７	…	１２６	１２７

2.13.2 １チャンネルＤ／Ａ出力デバイスドライバブロック

１チャンネルＤ／Ａ出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のＤ／Ａへ出力します。

このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.13-2　TM32DA D/A output (S)



- D/A channel number (0 to 127)

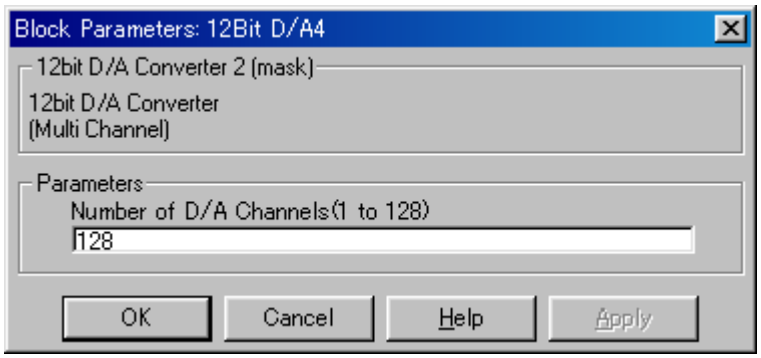
  出力するＤ／Ａのチャンネル番号を０～１２７の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.13.1（131ページ）を参照してください。

2.13.3 多チャンネルD／A出力デバイスドライバブロック

多チャンネルD／A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD／Aへ一括して出力します。出力の対象となるD／Aチャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のD／Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.13-3　TM32DA D/A output (M)



- Number of D/A channels (1 to 128)

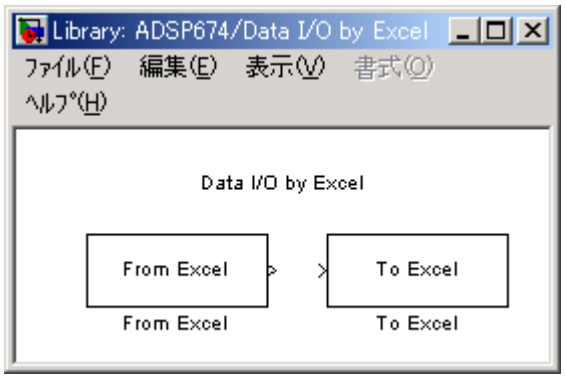
  出力するD／Aのチャンネル数を１～１２８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.13.1（131ページ）を参照してください。

2.14Data I/O By Excel ブロック

2.14.1概要

図 2.14-1 に、Data I/O By Excel用ウィンドウを開いた所を示します。

図 2.14-1　Data I/O By Excel ウィンドウ



この中には自動テスト機能で使用可能な From Excel/To Excel ブロックが登録されています。

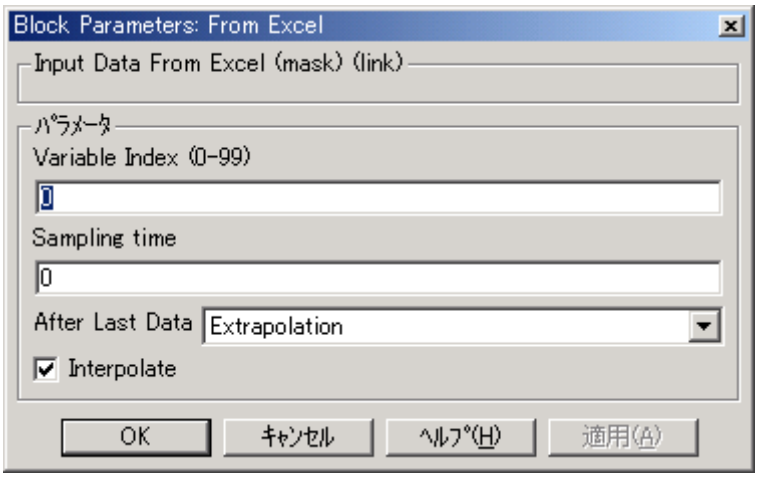
- From Excel　　MicroSoft Excel にて作成したデータを実行時動的にブロックへ取り込みます。
- To Excel　　MicroSoft Excel ブックへ実行時に動的にデータを出力します。

これらのブロックは、自動テストにて指定項目を設定し、実行した場合のみ有効となります。

2.14.2 From Excel ブロック

From Excel ブロックは、Microsoft 社製 Excel にて作成した、時間付データを実行時に動的にアクセスしモデルへデータを取り込みます。このブロックは自動テストにてモデルを実行した場合とＲＴＭＯＮにてＲＴオプションで“Frmxls ブロック使用”を指定し実行した場合のみ有効となります。又実行前にデータ領域をＤＳＰ上に確保します。ＤＳＰ上に必要なメモリが確保できない場合は使用できません。

図 2.14-1　From Excel



- Variable Index(0-99)
  自動テスト用の Excel ブックで指定する変数 Index を０～９９の範囲で指定します。ToExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。
- Sampling time
  データを取り込むサンプリングタイムを指定します。0 指定時はモデルのステップサイズに依存します。
- After Last Data
  指定した時間以降のデータ実装方法を指定します。
  Extrapolation：直前のデータ内容よりデータを外挿します。
  Setting To Zero:ゼロデータを出力します。
  Holding FinalValue:指定最終データを出力します。
- Interpolate
  データの指定範囲内でモデルのステップタイムに該当するデータがない場合の実装方法を指定します。
  チェックあり：データは内挿されます。
  チェックなし：取り込んだデータの最近のデータ値を出力します。

＜データ取り込み例＞

1） 最終データ後の出力データ: Extrapolation、Interpolate ： チェックあり 指定時

① Excel データ)

time	1	2	3	5	8
data	0	1	5	50	100

↓

ステップサイズ１Sec での取り込みデータ)

time	0	1	2	3	4	5	6	7	8	9	10	11
data		1			50	6	3.33		100	1	3.33	150
	補間			内挿		内挿			外挿			

② Excel データ)

time	0	1	2	2	3
data	0	0	1	5	10

↓

ステップサイズ１Sec での取り込みデータ)

time	0	1	2	3	4
data	0	0	5	10	15

＊Excel データが同じタイムステップで複数存在する時は、最後のデータが使用されます。

＊最終データが同じタイムステップで複数存在する時は Extrapolation が保障されませんので、注意してください。

2） 最終データ後の出力データ: Setting To Zero、Interpolate ： チェックなし 指定時

① Excel データ)

time	1	2	3	5	8
data	0	1	5	50	100

↓

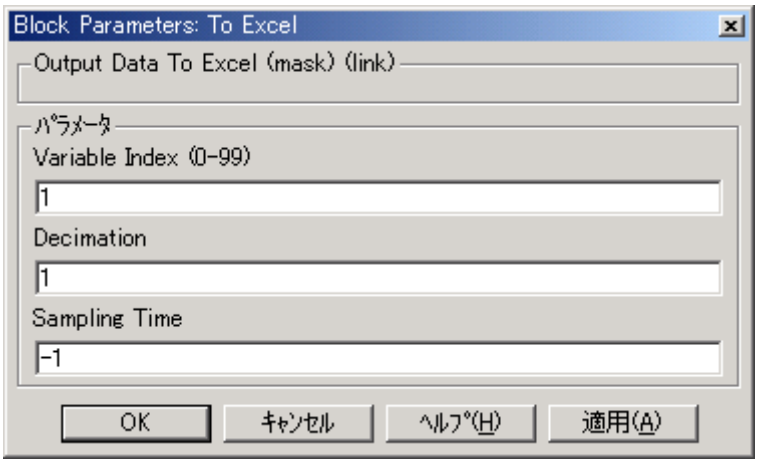
ステップサイズ１Sec での取り込みデータ)

time	0	1	2	3	4	5	6	7	8	9	10	11
data	0	0	1	5	5	50	50	50	100	0	0	0
										Setting To Zero		

2.14.3 To Excel ブロック

To Excel ブロックは、指定された Microsoft 社製 Excel ブックのシートへ、時間と入力データを書き込みます。このブロックは自動テストにてモデルを実行した時のみ有効となります。又実行前にデータ領域をＤＳＰ上に確保します。ＤＳＰ上に必要なメモリが確保できない場合は使用できません。

図 2.14-2　To Excel



- Variable Index(0-99)
  自動テスト用の Excel ブックで指定する変数 Index を０～９９の範囲で指定します。FrmExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。

- Decimation
  書き込みデータの間引きを指定します。１指定時は間引き無し、２指定時は、２データに対し１データ書込みとなります。

- Sampling Time
  データを書き込みサンプリングタイムを指定します。

3. 他のＤＳＰとの互換性

3.1 他の DSP との互換性

他の DSP 用の環境「ＡＤＳＰ３２４－４１B」「ＡＤＳＰ４０４－２４１Ｂ」（共にバージョン 4.x）で作成したブロック線図は、一部のブロックを除きコードを再生成する事で「ＡＤＳＰ６７４－３４１」での動作が可能です。

図 3.1-1　異種ＤＳＰ用環境間のＩ／Ｏデバイスの互換性

ブロックライブラリ	互換性
ADSP32X-00/50 ブロック	無し
ADSP404-00 ブロック	無し
ADSP32X-03/53 ブロック	有り
ADSP32X-06/56 ブロック	有り
ADSP32X-11/61 ブロック	有り
ADSP32X-13/63 ブロック	有り
ADSP324-141 ブロック	有り
ADSP324-143 ブロック	有り
ADSP324-145 ブロック	有り
TM32DA ブロック	有り

ＡＤＳＰ６７４－３４１
ＭＬ5.ｘ用Ｉ／Ｏライブラリ

第１版　2000.06.28
第１０版　2006.02.01

中部電機株式会社 ＣＳ事業部
TEL　0532-61-9566
FAX　0532-63-1081

ユーザー定義ブロック

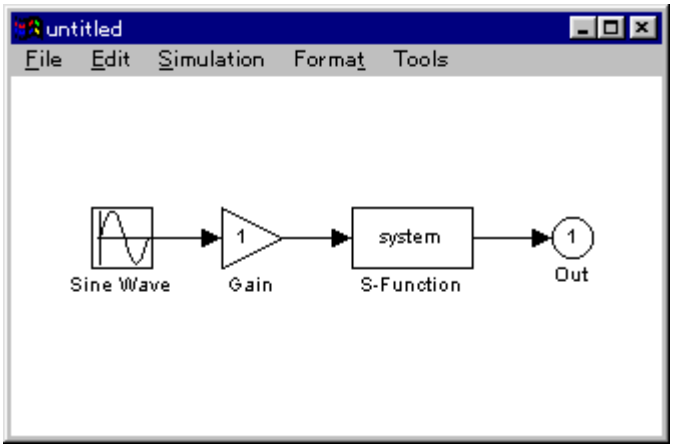
＜＜　目次　＞＞

1　概要 2
2　利用出来るＳ－Ｆｕｎｃｔｉｏｎ 2
3　ユーザ定義ブロックの作成 3
3.1　ＭＡＴＬＡＢ　５．ｘの場合 3
3.1.1　概略手順 3
3.1.2　ソースファイルの作成 3
3.1.3　ＤＬＬの作成 5
3.1.4　ＳＩＭＵＬＩＮＫでの動作確認 5
3.1.5　ライブラリの作成 5
3.1.6　RTW でのコード生成 7
3.1.7　RTW での動作確認 7

1　概要

ここでは、ＳＩＭＵＬＩＮＫの拡張機能である S-Function を用いて作成されているブロックを用いたブロック線図の実時間コード化について説明します。

S-Function はＳＩＭＵＬＩＮＫの拡張機能の一つで、ブロック線図で使用出来る、カスタマイズされたブロックを作成する機能です。出来たブロックは下のブロック線図の様に標準のブロックと接続して、ブロック線図の一部として使用します。



2　利用出来るＳ－Ｆｕｎｃｔｉｏｎ

ＳＩＭＵＬＩＮＫでは、色々な言語による S-Function が利用出来ましたが、Real－TimeWorkshop を用いてＤＳＰ用のＲＴモデルを作成する場合にはＣ言語による S-function しか利用出来ません。

＜S-Function で利用可能な言語＞

記述言語	ＳＩMULＩＮＫ	Real-Time Workshop
MATLAB 言語(M ﾌｧｲﾙ)	可	不可
Ｃ言語	可	可
ＦＯＲＴＲＡＮ言語	可	不可

これは専ら、ＤＳＰ用のコンパイラが、Ｃコンパイラしか無い事に由来しています。
従って、将来 Real-Time Workshop を用いてＤＳＰ用実時間コードを作成する予定がある場合、S-Function はＣ言語で作成する必要があります。

3　ユーザ定義ブロックの作成

3.1　　ＭＡＴＬＡＢ　５．ｘの場合

3.1.1　　概略手順

ＤＳＰを用いた実時間モデルで使える S-Function によるユーザ定義ブロックの作成手順は以下の通りとなります。

１、ソースファイルの作成

２、ＤＬＬの作成

３、ＳＩＭＵＬＩＮＫでの動作確認

４、ライブラリの作成

５、ＲＴＷでの動作確認

3.1.2　　ソースファイルの作成

ソースファイルは、汎用のテキストエディタにて作成します。

ＳＩＭＵＬＩＮＫには、S-Function のサンプルとして CSFUNC が用意されています。本製品ではこれに倣ったサンプル、CSFUNC.C が以下のフォルダーに用意されています。

MATLAB¥RTW¥C¥ADSP32X¥SAMPLE¥CSFUNC¥BLOCK

これを参考にすると良いでしょう。隣りの..¥model フォルダーにある、Usersfcn.mdl はこの S-Function を使用するブロック線図の例です。

このソースファイルはＭＡＴＬＡＢの版によってはＤＳＰのコンパイラに適合する様に一部変更されています。変更前のソースファイルが、変更されている場合は同じフォルダーに CSFUNC.ORG という名称で格納されています。併せご参照ください。又、この関数及びソースファイルの詳しい内容等、及び S-Function の詳しい説明はＳＩＭＵＬＩＮＫのユーザーズガイドをご参照ください。

Using SIMULINK Version 2 の８章 「S-Function」が参考になります。

作成する S-Function のソースファイル名には次に示す制約があります。

- 名称の先頭は、英字で始めてください。
  - 良い例： a3bc.c
  - 悪い例： 3abc.c （理由＝数字で始まっている）
- 名称の文字数は、８文字以内としてください。
  - 良い例：goodname.c
  - 悪い例：nameislonger.c
- 名称で使う文字は英数字と __ 記号のみとしてください。
  - 良い例：a32_abc.c
  - 悪い例：a(#1).mdl　（理由＝かっこ、#等の記号を使っている）
- 英字の大文字と小文字は同じものとして取り扱われます。

作成するS-FunctionはＳＩMULＩＮＫ用のＤＬＬと、ＤＳＰ用のライブラリの２つが必要となりますが、場合によっては、例えば希望とする処理が数値計算等メモリー内のデータ操作だけで終わってしまう様な場合、両者は、同一のソースファイルを使う事ができます。
しかしそうでなく、例えばＤＳＰのＩ／Ｏを扱う様な場合は、ＤＬＬは（ＰＣ上で動作し、ＤＳＰのＩＯは直接扱えませんから）ダミーとなり、ＤＳＰ用ライブラリになった時点で初めて処理実体が必要となります。この様な場合は、同一ソースファイルでありながら、部分部分でＤＬＬ用にコンパイルしている時とＤＳＰ用にコンパイルしている時と異なる記述が必要となります。そういった場合には、次の条件付きコンパイルを用いると同一ソースファイルとする事が可能です。

```
#ifdef    MATLAB_MEX_FILE
          /* SIMULINK 用 DLL の時に有効となります。  */
#else
          /*  ＤＳＰ用ライブラリの時に有効となります。  */
#endif
```

3.1.3　　ＤＬＬの作成

ＭＡＴＬＡＢ　５．ｘでは、ユーザ定義 S-Function を含むブロック線図から Real-Time Workshop で実時間コードを作成する場合には、必ず、同 S-Function のＤＬＬ版を用意する必要があります。ＳＩＭＵＬＩＮＫ上ではMファイル版の S-Function でも動作しますが、Real-Time Workshop によるコード生成は、ＤＬＬが無いと正しく行えません。
ＤＬＬによる S-Function の作り方についてはＳＩＭＵＬＩＮＫのユーザーズマニュアルを参照してください。以下が参考になります。
MATLAB Application Program Interface Guide の２章 「Creating C Language MEX-Files」
MATLAB Command window から help mex 又は mex -help を実行

3.1.4　　ＳＩＭＵＬＩＮＫでの動作確認

ＤＬＬによる S-Function が出来たらＳＩＭＵＬＩＮＫ上で正しく動作するか確認します。作成したＤＬＬはカレント又は PATH の通ったフォルダーに格納します。一度 SIMULINK で動作させた後、変更等で再度　mex を行う場合は、MATLAB を再起動してください。

3.1.5　　ライブラリの作成

ＳＩＭＵＬＩＮＫ上で正しく動作する事が確認出来たら、ＤＳＰ用ライブラリを作成します。

ＲＴモデル用ライブラリ化は、以下の手順で行います。

（１）ライブラリ生成用ディレクトリを作成し、ここにＭＡＴＬＡＢカレントディレクトリを移動してください。（ＭＡＴＬＡＢのＣＤコマンド参照）

（２）カレントディレクトリに、S-Function のソースファイルをコピーします。カレントディレクトリには、この S-Function のソース以外のＣソース（拡張子 .c ）を格納しないでください。誤ってライブラリ化されてしまいます。

（３）ライブラリの作成は、ライブラリ作成コマンド genlib674 で行います。
標準的な使い方は、genlib674　-r　-q　です。これによりＣソースがコンパイルされ、opt_lib.lib という名称のライブラリが作成され、所定のディレクトリにコピーされます。

（４）ライブラリを複数作成する場合は、 genlib674　-r　-q　mylib 等とすると、任意の名称のライブラリ mylib.lib が作成されます。 但し、a674_rt 及び libsrc67 は予約となっていてユーザは利用出来ません。-r はライブラリを所定のディレクトリに自動的にコピーする指示です。この指示をしない場合、カレントディレクトリにライブラリが作成されていますので、後でコピーしてください。-q は -r の指示により自動コピーする際に、以前に作成した同名ライブラリが有っても強制的に上書きします。 -q が無いと上書きするか否か問い合わせてきます。

（５）genlib674 -help とするとオンラインヘルプが参照できます。併せ参照してください。出来たライブラリをコピーする先のディレクトリ名もこれで確認できます。

（６）mex で S-Function の DLL コード生成が正しく出来ても、genlib674 によるコンパイルでエラーが出る場合があります。この理由の殆どは、Ｃコンパイラのエラーチェック仕様の相違によるものです。ＤＳＰ用Ｃコンパイラの方が（必要以上に）型チェックが厳しい為、ＭＥＸの DLL で正常にコンパイル出来て、正常に動作しているにも拘わらず、エラーが出る場合があります。

その最大のものが、real_T * と double * の相違です。real_T は#define により double として定義されていますが、ＤＳＰ用コンパイラは別物として扱ってしまいます。例えば simstruc.h で、関数プロトタイプ宣言の引数が real_T * と定義されていて、実際の関数の引数が double * となっていると、エラーとなってしまいます。

SIMULINK に標準で添付されている S-Function の例 csfunc.c でもこの現象が出ます。これは以下の通り（４ヵ所）変更すれば、問題無くコンパイルできます。

```
変更前　　static void mdlInitializeConditions(double *x0, SimStruct *S)
変更後　　static void mdlInitializeConditions(real_T *x0, SimStruct *S)

変更前　　static void mdlOutputs(double *y, double *x, double *u,
                                                    SimStruct *S, int tid)
変更後　　static void mdlOutputs(real_T *y, const real_T *x,
                                    const real_T *u, SimStruct *S, int tid)

変更前　　static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
変更後　　static void mdlUpdate(real_T *x, const real_T *u, SimStruct *S, int tid)

変更前　　static void mdlDerivatives(double *dx, double *x, double *u,
                                                    SimStruct *S, int tid)
変更後　　static void mdlDerivatives(real_T *dx, const real_T *x,
                                    const real_T *u, SimStruct *S, int tid)
```

（上記変更は添付されている CSFUNC.C では訂正されています。）

（７）ADSP32X シリーズと、ADSP674 とは同じソースファイルを用いてライブラリ化できますが、ライブラリそのものは共用出来ません。同じフォルダーで genlib32 と genlib674 をそれぞれ行い、個別にライブラリの作成を行ってください。この時ライブラリ名称を同じとすると、カレントディレクトリ上でライブラリが上書きされますので、生成する都度、それぞれ所定のフォルダーにコピーしてください。

注意）上記内容は Small モデルを基準に説明しています。Large モデルで作成時は geblib674b にて作成してください。又モデルによりライブラリが格納されるディレクトリが以下のように異なります。

Small モデル　　MATLAB ﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥lib674

Large モデル　　MATLAB ﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥lib674¥big

3.1.6　　RTW でのコード生成

ライブラリ化が終わり、作成されたライブラリが所定のフォルダに格納された後、再度 Real-Time Workshop 実時間コードを生成します。これにより先程作成されたライブラリは自動的にリンクされます。

3.1.7　　RTW での動作確認

Real-Time Workshop で生成した実時間モデルは、リアルタイムモニターにて実行してみて、要所要所の波形を見る（或いはロギングして見る）のが唯一の方法です。
ＤＳＰ用ライブラリになってからの、S-Function 単体のデバッグは一般には困難です。
もし S-Function（又はその内部）のデバッグを行う必要があるなら、ソースファイルにパッチを当てる事で動作結果をメモリー（大域変数）に残す様に変更し、リアルタイムモニターにて動作後、弊社付属ソフトウェア（ＡＤＳＰ４０Ｘユーティリティー）にてメモリーを表示する方法が使えます。メモリーアドレスは、モデルと同じディレクトリにマップファイル *model*.MAP が生成されていますので、これを参照してください。

ユーザ定義ブロック

第１版　2000.06.28
第３版　2003.06.11

中部電機株式会社 ＣＳ事業部
TEL　0532-61-9566
FAX　0532-63-1081

ユーザー定義関数

＜＜　目次　＞＞

1　概要 3

2　実行形態 4

2.1　処理順序 4

3　関数詳細説明 7

3.1　UserInitializeSizes() 7

3.2　UserInitialize() 8

3.3　UserOutput() 8

3.4　UserUpdate() 9

3.5　UserBackground() 9

3.6　UserTerminate() 9

4　データ交換 11

4.1　データ交換規則 11

4.2　入出力点数の自動整合 13

4.3　入出力受け渡しバッファ 14

5　ユーザ定義関数からのリアルタイムモデル停止 15

5.1　方法１　～　Stop Simulation 15

5.2　方法２　～　ｼﾐｭﾚｰｼｮﾝ停止フラグの操作 15

6　使用方法 16

6.1　USERFUNC.C のインストール 16

6.2　USERFUNC.C のコンパイル 16

6.3　USERFUNC.C のコンパイルとリンク 18

6.4　USERFUNC.C のデバッグ 19

6.5　個別モデル専用のユーザ定義関数 20

6.6　ＭＬ4.2c とＭＬ5.1 の共用 20

7　制限事項 21

7.1　動作上の機能制限 21

7.2　使用出来ないソースファイル名 21

7.2.1　MATLAB 4.2c の場合 21

7.2.2　MATLAB 5.1 の場合 23

7.3　長時間に渡るＤＳＰの専有 24

7.4　長時間に渡る割り込み禁止 25

1　概要

本章は、ＭＡＴＬＡＢ対応ライブラリ「ADSP324-41B」「ADSP404-241B」「ADSP674-341」(Rev 4.x 版以後）於いて、リアルタイムモデルにユーザ定義関数を組み込む方法について説明しています。

この機能は SIMULINK では記述しづらい手続きや、より高速化／高機能化等を実現したい場合、ユーザのＣ言語によるソフトウェア資産を活用したい場合等に、ユーザが記述したＣ言語の関数を SIMULNIK ブロック線図で記述した処理に組み込む事により、両者を並行して実行する為ものです。予め定められた関数名と大域変数を用いる事によりブロック線図とのインターフェースを取り、データを交換する事もできます。

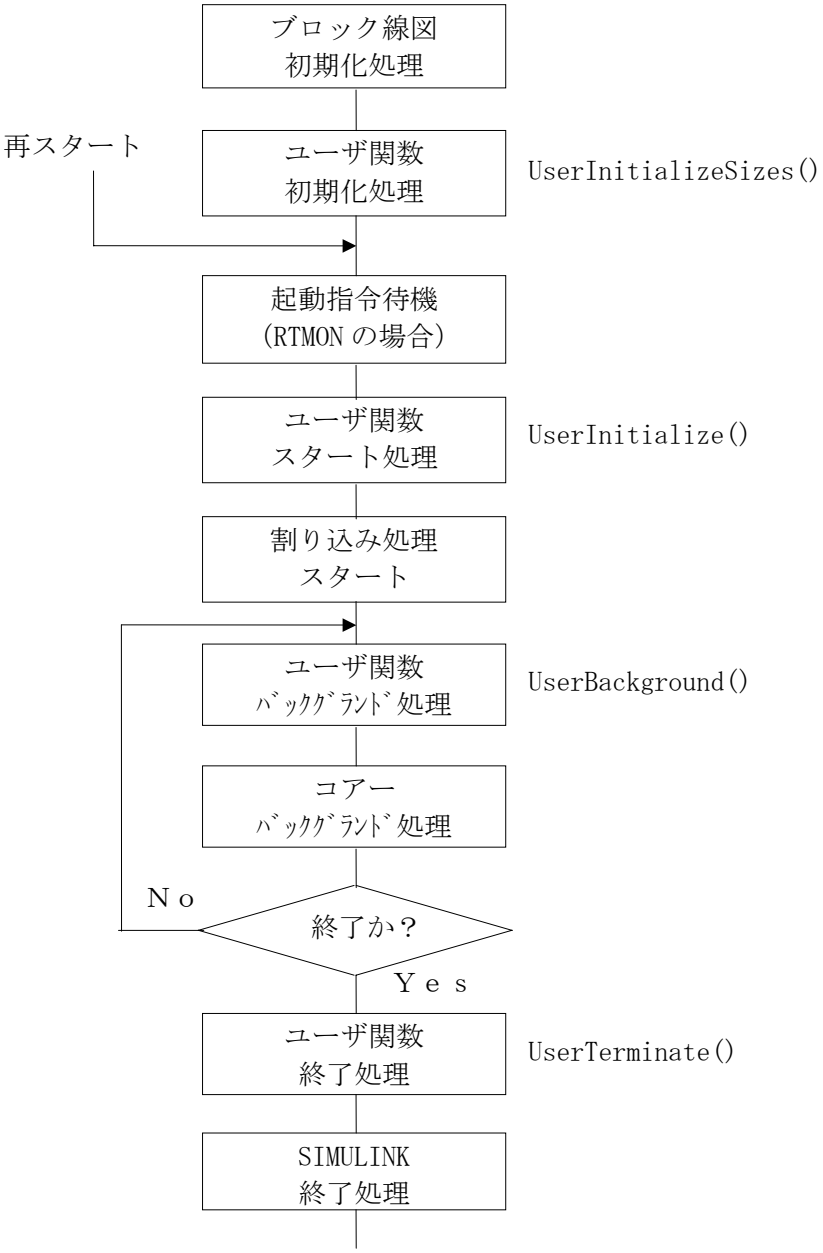
2　実行形態

ここでは、SIMULINK ブロック線図の処理とユーザ定義関数の処理との順序関係、データ交換の方法等を説明します。

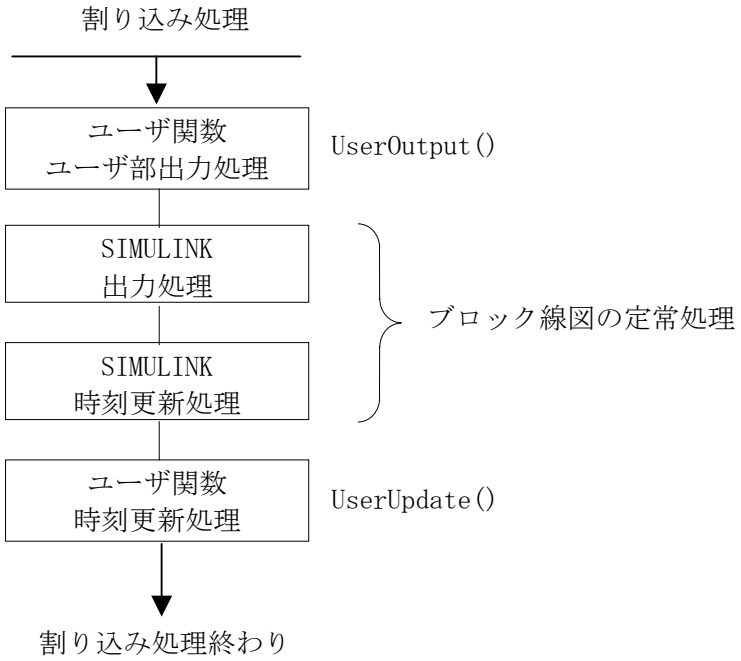
2.1　　処理順序

処理の順番は以下のフローチャートにより行なわれます。

- メインルーチン



- 割り込み処理ルーチン

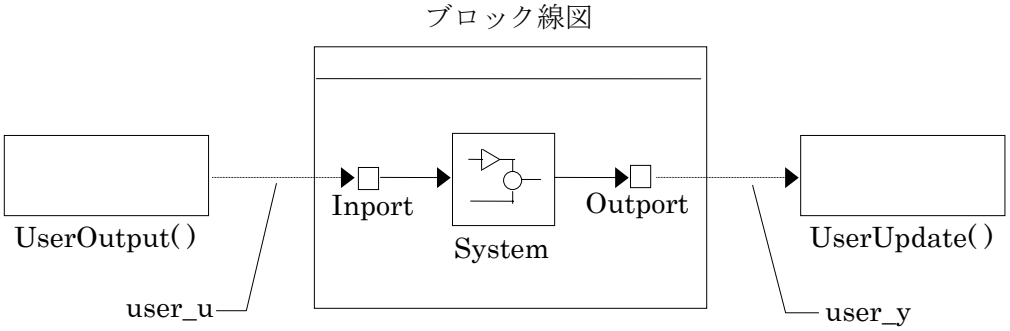


メインルーチンは、初期化・起動司令待機後、割り込みを起動したらバックグランド処理を行います。ここでは通常、
１、リアルタイムモデルが終了時刻に達したか確認を行う。
２、ＲＴＭＯＮ、又は、ＵＩＬからのランタイムアクセスを処理する。
３、それ以外のバックグランド処理を行う。
以上の作業を行っています。

一方、割り込み処理は、実行形式によりタイマー又は外部割り込みにより駆動され、リアルタイムモデルの１ステップ分の計算を行い、メインルーチンへ戻ります。

フローチャートに示す通り、ユーザ定義関数では、これらの処理の要所々々でユーザの定義する関数を呼び出すことが出来ます。

別の見方として、SIMULINK ブロック線図的な表記では下記の様な接続関係となります。



中央が SIMULINK のブロック線図で、両横の [　　] がユーザが定義した処理です。
処理順序は図の向かって左側から右側へ順に処理されます。破線矢印（ ⟶ ）は大域変数を使用したデータを転送を表しています。ブロック線図には現れません。このように、ユーザ処理と SIMULINK ブロック線図との間のデータ交換は SIMULINK のトップレベルの Inport、Outport を通じて行います。
user_u,user_y と Inport,Outport との対応の詳細については“データ交換”の章を参照してください。

3　関数詳細説明

ユーザ定義の関数は予め予約された関数名が６つ用意されています。

UserInitializeSizes()	ユーザ固有の初期化処理を行います。また、ユーザ関数とSIMULINK とのデータ交換に使う領域もここで確保します。
UserInitialize()	システムが機動・停止を繰り返す必用がある場合、再起動に必用な初期化処理を行います。
UserOutput()	ユーザブロックから SIMULINK への出力を計算します。
UserUpdate()	SIMULINK の出力を受け取り処理すると共に時刻更新の処理を行います。
UserBackground()	ユーザのバックグランド処理を行います。
UserTerminate()	システム停止の処理を行います。

上記関数はテンプレートファイル　USERFUNC.C　に予め定義されており、最低限必要な処理が予め記述されています。ユーザはこれに必用な手続きを書き足します。 関数名が定められている事を除けば、通常の関数と同じです。

3.1　UserInitializeSizes()

関数定義　　int　 UserInitializeSizes(int in_num, int out_num);

意味　　この関数はモデル起動直後に１度だけ呼び出されます。システムの初期化、作業領域の確保等の初期化作業のうち最初に１度だけ行えばよい処理をここで行なってください。

引数　int　in_num;

SIMULINK ブロック線図への入力ポート（Inport）の数が与えられます。データ受け渡しに使用する大域変数配列 user_u は、最低限これ以上の長さが必用です。この変数は同時に、他のユーザ定義関数から参照可能とする為、u_num 等の大域変数に保存する事をお勧めします。これにより user_u を、確保した長さ以上にアクセスしないよう管理可能です。

int　out_num;

SIMULINK ブロック線図の出力ポート（Outport）の数が与えられます。データ受け渡しに使用する大域変数配列 user_y は、最低限これ以上の長さが必用です。user_y は、ユーザの使用の有無に関わらず、out_num の点数分書き込みが行われます。この変数は同時に、他のユーザ定義関数から参照可能とする為、y_num 等の大域変数に保存する事をお勧めします。これにより user_y を、確保した長さ以上にアクセスしないよう管理可能です。

戻り値　　エラーの有無を返します。正常終了で０を、それ以外は０以外を返します。エラーを返すとメインルーチンは処理を中断します。リアルタイムモニターでは、このエラーをヒープ領域不足として扱い、ステータス表示します。ここでユーザがエラーを返す可能性がある場合、表示の意味に注意してください。

3.2　UserInitialize()

関数定義　　void　UserInitialize(void);

意味　　システムが起動停止を繰り返すような場合、起動ごとに必用となる初期化処理をここに記述します。

引数　　ありません。

戻り値　　ありません。

3.3　UserOutput()

関数定義　　void　UserOutput(double　t);

意味　　定常状態でステップサイズ毎に、最初に呼び出される処理です。ユーザ処理部分の出力、(即ち SIMULINK ブロック線図への入力) を決定します。

引数

double　t;

現在の時刻が与えられます。システム起動後最初の呼び出しには、リアルタイムモニター、又は SIMULINK の Real-Time Options で指示した開始時刻 (Start time) が与えられます。通常は０です。その後、呼び出し毎にステップサイズだけ加算されます。

戻り値　　ありません。

3.4 UserUpdate()

関数定義　　void　UserUpdate(void);

意味　　定常状態でステップサイズ毎に、SIMULINK ブロック線図の計算が終了した直後に呼び出される処理です。必用に応じて SIMULINK ブロック線図の出力を受け取り、処理をします。また、時刻更新の為の処理もここで行います。

引数　　ありません。

戻り値　　ありません。

3.5 UserBackground()

関数定義　　void　UserBackground (void);

意味　　ユーザのバックグランド処理を記述します。通常メインルーチンでのバックグランド処理は、割り込みによるリアルタイムモデルの計算を待ち受ける傍ら、ＲＴＭＯＮ，ＵＩＬ等によるランタイムアクセス（実行時のパラメータ変更）をソフト的なハンドリングで行っていますので、これらを妨げないように、時分割で処理する必要があります。長時間かかりそうな処理は、何回かに分け、この関数が呼び出される毎に順番に一部分づつ実行するようにしないといけません。詳しくは、7.3 長時間に渡るＤＳＰの専有 を参照してください。

引数　　ありません。

戻り値　　ありません。

3.6 UserTerminate()

関数定義　　void　UserTerminate(void);

意味　　システムの終了処理を記述します。リアルタイムモデルが停止する直前に１度だけ呼び出されます。リアルタイムモデルは以下の条件で停止します。

１、 現在時刻が終了時刻（Stop time）に到達した場合。

２、 ブロック線図の Stop Simulation ブロックの入力が０以外になった場合。

３、 リアルタイムモニターの停止ボタンが押された場合。

４、 ユーザプログラムから、ＵＩＬ経由でリアルタイムモデルの停止機能が呼び出された場合。

５、 ユーザ定義関数で、シミュレーション停止フラグを操作した場合。

ＤＳＰボード付属コマンドの RESET324、HOLD324 等でＤＳＰを停止した場合は、この関数は呼び出されません。

引数　　ありません。
戻り値　　ありません。

4　データ交換

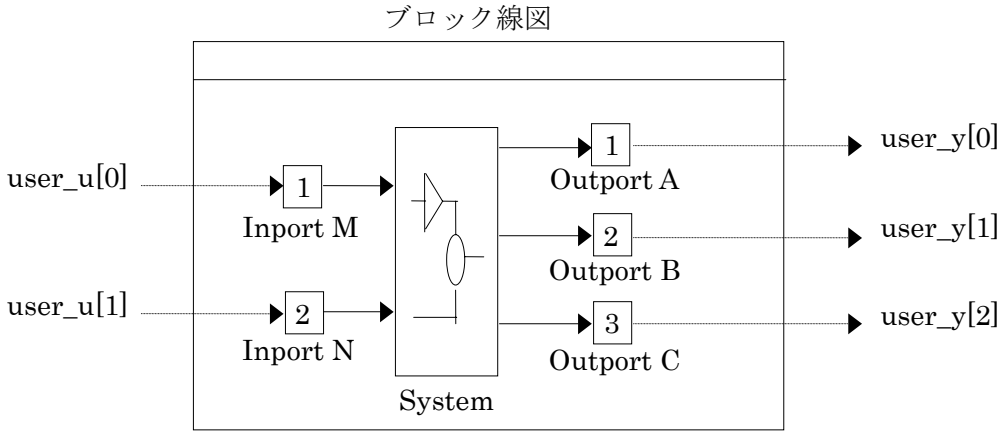
4.1　　データ交換規則

ユーザ定義処理と SIMULINK ブロック線図との間は大域変数を経由してデータ交換をします。データ領域は USERFUNC.C に大域変数として定義されており、実体は初期化関数 UserInitializeSizes() 内で malloc() にて割り付けます。

double	*user_u;	SIMULNI ブロック線図への入力データを格納します。
double	*user_y;	SIMULNI ブロック線図からの出力データが格納されます。
int	u_num;	user_u の長さを記憶するのに使用します。
int	y_num;	user_y の長さを記憶するのに使用します。

user_u、user_y の長さはそれぞれ u_num、y_num で管理し、確保した長さ以上にアクセスしないように注意してください。user_u[ ](user_y[ ])の添え字の最大は、u_num-1(y_num-1)となります。

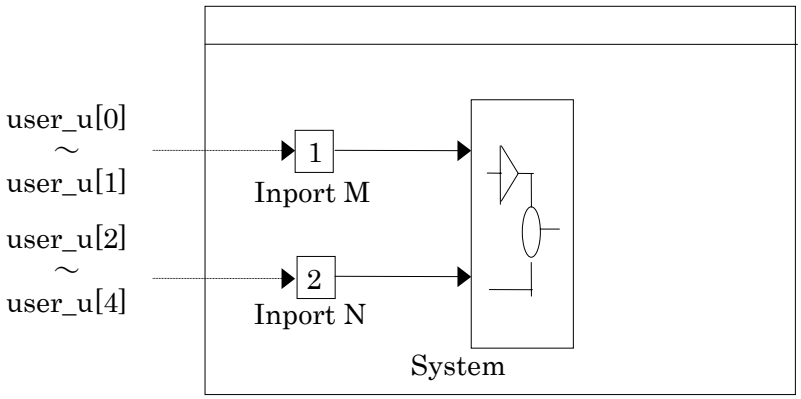
user_u、user_y と SIMULINK ブロック線図の Inport、Outport との対応は下記に示す通りです。この場合は、Inport、Outport はいずれもスカラーとします。



この例のように、SIMULINK ブロック線図のポート番号（□の中の番号）－１が配列の添え字となります。この場合、u_num == 2、y_num == 3 となります。

Inport、Outport がベクトルの場合の例を次に示します。

ブロック線図



1 本のベクトルは配列の中の連続する変数に割り当てられ、ベクトルの要素が若い方が配列の添え字の若い方に相当します。次の Inport はそのすぐ後の変数に相当します。
例えば、このブロック線図で、Inport M が幅２、Inport N が幅３のベクトルであるとすると、以下の様に対応します。

user_u[0]　　Input M（１）
user_u[1]　　Input M（２）
user_u[2]　　Input N（１）
user_u[3]　　Input N（２）
user_u[4]　　Input N（３）

4.2　　入出力点数の自動整合

ユーザ定義関数でブロック線図との間でデータ交換をする、入出力ブロックの点数（user_u[ ] の添え字）は、ブロック線図とユーザ定義関数とで整合していないといけません。特に、ブロック線図で定義された以上の長さについてアクセスをすると、場合によってはＤＳＰが暴走する等の不都合が生じる場合があります。

ＭＡＴＬＡＢ　５．１では、この点数を表す定数がヘッダに定義されており、これを参照する事により、コンパイル時点でエラーを発生させてユーザに警告する事が可能です。
例えばユーザ定義関数側が出力（ユーザ出力→ブロック線図）として ２つのポートを、入力（ブロック線図→ユーザ時刻更新）として３つのポートを、それぞれ想定してコードを作成した場合、*MODEL*.UFN コードの適当なヵ所に、

```
#if   ML51
#if        NINPUTS != 2
 #error：Not expected number of InputPort of Model.
#endif
#if        NOUTPUTS != 3
 #error：Not expected number of OutputPort of Model.
#endif
#endif
```

と記述します。
NINPUTS がブロック線図への入力、NOUTPUTS がブロック線図からの出力の点数を表しています。点数とはブロックの数でなく、各ブロックのベクトルの、幅の合計ですので注意が必要です。入力ブロックが１個でも、ここへ接続されている信号が幅３のベクトルの場合 NINPUTS は 3 となります。
もし､ブロック線図に想定外の点数の入出力ポートが定義されると、Not expected number … が MATLAB Command window に表示され、コード生成作業が中断されます。

また、この定数を参照する事により、自動的に入出力点数を整合させる事も可能です。コンパイル時点で点数が判明していますので、作業用変数領域の確保が

```
double   work[NINPUTS];
```

等として的確に確保する事ができます。

4.3　　入出力受け渡しバッファ

ＭＡＴＬＡＢ5.1 の場合は、ＭＡＴＬＡＢ 4.2c 版の場合と同じく、user_u、user_y でも受け渡しができますが、入出力ポートを定義した大域変数が用意されており、これを直接参照する事もできます。入力ブロック、出力ブロックを、それぞれ構造体で定義しています。構造体定義は *MODEL*.H にあり、実体は *MODEL*.PRM に記述されています。

＜例：*MODEL*.H　（抜粋）＞

```
/*
 * External Inputs Structure
 *
 * Note: Individual field names are derived from the block name.
 *
 */

typedef struct ExternalInputs_tag {
   real_T In[2];
} ExternalInputs;


/*
 * External Outputs Structure
 *
 * Note: Individual field names are derived from the outport block names
 *
 */

typedef struct ExternalOutputs_tag {
   real_T Out[3];
} ExternalOutputs;
```

＜例：*MODEL*.PRM　（抜粋）＞

```
/* External Inputs Structure */
ExternalInputs rtU;

/* External Outputs Structure */
ExternalOutputs rtY;
```

5　ユーザ定義関数からのリアルタイムモデル停止

ユーザ定義関数からリアルタイムモデルを安全に強制停止させるには、以下の方法を用いてください。何れの方法も、停止要求が受理された後、ブロック線図の停止処理、ユーザ定義関数の停止処理等が呼び出され、ＲＴモデルが停止します。
これ以外の方法（例えばタイマーを強制的に止める等）で停止させた場合、本来行うべき終了処理 （Terminate）が呼びだされません。

5.1　　方法１ ～ Stop Simulation

ユーザ定義関数の出力 user_u を使って、停止制御用信号をブロック線図に渡します。ブロック線図ではこれを Stop Simulation ブロックに入力します。
常時は、０を出力しておき、停止させたい場合は１を渡します。

この方法は、停止する直前にブロック線図でもこれを参照して、何がしかの処理が行えますので、ブロック線図内で停止直前に行うべき処理がある場合に利用できます。

5.2　　方法２ ～ ｼﾐｭﾚｰｼｮﾝ停止フラグの操作

メイン関数に対し、シミュレーションを停止させる要求を渡すフラグが用意されています。これを直接操作します。

ＭＡＴＬＡＢ 4.2c の場合

フラグ	extern char stop_execution_flag;
停止指示	stop_execution_flag = 1;

ＭＡＴＬＡＢ 5.1 の場合

フラグ	extern SimStruct *const rtS;
停止指示	ssSetStopRequested(rtS,1);

この方法はブロック線図に頼らず強制的に停止が可能です。

6　使用方法

6.1　　USERFUNC.C のインストール

本製品のインストール直後の状態では、ユーザ定義関数は使用しない状態になっています。これを使用出来るようにするには、以下の作業を行います。

ＭＡＴＬＡＢをインストールしたフォルダ（%MATLAB_ROOT%）以下の,
次に示すフォルダに USERFUNC.C が格納されています。これをＭＡＴＬＡＢのカレントディレクトリにコピーします。

- ＭＡＴＬＡＢ 4.2c の場合

%MATLAB_ROOT%¥Codegen¥Rt¥Chubu¥Os¥

- ＭＡＴＬＡＢ 5.1 の場合

%MATLAB_ROOT%¥Rtw¥c¥Adsp32x¥Rt¥　　(ADSP324-41B の場合)
%MATLAB_ROOT%¥Rtw¥c¥Adsp32x¥Rt404¥　　(ADSP404-241B の場合)
%MATLAB_ROOT%¥Rtw¥c¥Adsp32x¥Rt674¥　　(ADSP674-341 の場合)

その後､ＭＡＴＬＡＢのカレントディレクトリの USERFUNC.C を必要に応じて変更してください。

6.2　　USERFUNC.C のコンパイル

ユーザー定義関数のファイル“USERFUNC.C”は、上記変更により、Buildボタンから自動的にコンパイルされますので、特にユーザが個別にコンパイルする必要はありません。しかし一般には、プログラマーが記述したソースファイルは色々な意味でエラーを含んでおり、書きっぱなしでそのまま正常に動作するのは稀なのが通常です。当然、１つでもエラーがあると実行ファイルは生成されません。その度にUSERFUNC.Cを修正し再度Buildをするのは、SIMULINKモデルのコード生成とコンパイルをも毎回行なう事になり、その分、無駄な時間を費やす事になります。
そこで、最低限タイプミスや文法上の誤りを USERFUNC.C から予め取り除いておく事をお勧めします。

コンパイルの方法は以下の通りです。

1. ＭＳ－ＤＯＳプロンプトを起動します。
2. ＤＯＳのカレントディレクトリをSIMULINKブロック線図が格納されているディレクトリ（Matlab Command Wondow のカレントディレクトリ）に移動します。
3. 以下のコマンドでコンパイルします。

```
cl30  userfunc.c                    (ADSP324-41Bの場合)
cl30 -v44  userfunc.c               (ADSP404-241Bの場合)
cl6x –mv6700  userfunc.c            (ADSP674-341の場合)
```

不幸にもエラーがあまりに多く、エラーメッセージが画面からスクロールアウトしてしまう場合は、

```
cl30  userfunc.c  >  error
```

等として、エラーメッセージをファイルに残すとよいでしょう。この様にすると画面に出力されていたエラーメッセージが、ファイル error に保存されます。
Ｃ言語ではちょっとしたタイプミス１文字でもそれが文法的なエラーだと、それを契機に連鎖的に多数のエラーが出る場合があります。これはコンパイラが最初のエラーにより後続の正しい部分をエラーと誤解してしまう為です。真のエラーの数とエラーメッセージの数は必ずしも一致しませんから、エラーメッセージが多い事自体は全く悲観する必要はありません。このような場合、えてして、最初に表示されたエラー１つを解決すれば（文法的には）全て解決します。従って、先頭に近い側のエラーから順に確認して行き、原因が判らない場合はもう一度コンパイルしてみます。
肝心の最初に表示されるエラー行をスクロールアウトで見失わない為には、エラーメッセージをファイル化するのは有効な方法です。

6.3　　USERFUNC.C のコンパイルとリンク

コンパイル段階でエラーが無くなってもリンクの段階でエラーが出たり、又は、ブロック線図は無変更でユーザ定義関数のみ変更した場合等で、ユーザ定義関数のコンパイルとリンクのみ行いたい場合があります。その場合はブロック線図のコード化とコンパイルは不要ですのでその時間が無駄となります。
その場合は以下の方法でユーザ定義関数のコンパイルとリンクが行えます。尚以下の説明はＭＡＴＬＡＢ 5.1 の例ですが、ＭＡＴＡＬＢ 4.2c にても同様の方法が可能です。

（１）ユーザ定義関数ソース（エラーがあっても構わない）がカレントディレクトリにある状態で、ＲＴＷでコード生成を行います。

（２）この時の、MATLAB Command Window に表示されたメッセージに注目します。例えば、test.mdl をコード生成した場合のメッセージは以下となります。

```
### Starting RTW build procedure for model: test
### Invoking Target Language Compiler on test.rtw
### Loading TMW TLC function libraries
### Initial pass through model to cache user defined code
### Creating source file test.c
### Creating part 1 of registration file test.reg
### Creating parameter file test.prm
### Creating model header file test.h
### Creating part 2 of registration file test.reg
### TLC code generation complete.
### test.mk which is generated from D:¥MATLAB.511¥rtw¥c¥adsp32x¥a32x.tmf is up to date
### Building test: nmake /nologo -f test.mk STEP=0.01 UFUNC=un
### Compiling test.c
### Compiling rt5sim.c
（以下省略）
```

（３）上記 ＿＿＿ 部分を元にして、バッチファイル MK.BAT をカレントディレクトリに作ります。この場合 MK.BAT の内容は以下となります。

```
nmake /nologo -f test.mk STEP=0.01 UFUNC=un
```

内容はメッセージの ＿＿＿ 部分そのままですので、コピー&ペーストを使うと便利です。

（４）コンパイル作業は、MATLAB Command Window で！mk とします。

6.4　　USERFUNC.C のデバッグ

一般にデバッグ作業は、弊社デバッガ、ＭＳＣアダプタ等の開発ツールの使用が基本となりますが、 SIMULINK ブロック線図ぐるみのデバッグはかなり大変で、お勧めできるものではありません。
できれば、テスト用のダミーの SIMULINK ブロック線図に代わる部分を作成しておき、これを相手にしてユーザ関数部分のみをデバッグしておき、最後に SIMULINK に組み込んで実行する方法をお勧めします。

その場合に使用するように、テストの為の main()関数とダミーの SIMULINK ブロック線図処理関数のテンプレートが USERFUNC.C に組み込まれています。
同ファイルの後半 #if　TEST_MAIN 以下最後までがその部分に該当します。この部分はファイル先頭のタイトルコメント直後の、#define　TEST_MAIN　0 により、通常は無効となっています。これを #define TEST_MAIN　1　と変更することにより、main()が有効となるようにしてご利用ください。

6.5　　個別モデル専用のユーザ定義関数

ユーザ定義関数ソースファイル userfunc.c は、同じディレクトリ内にある全てのモデルで共用となり、このディレクトリ内で作成される全ての実時間モデルに、同じ関数がリンクされます。

もし、モデル毎に異なるユーザ定義関数を用意したい場合は、ソースファイルの名称を標準の userfunc.c でなく、*model*.ufn とします。(*model* はブロック線図の名称と同じ) ファイルの内容は同じで、単にファイル名称を変更するだけです。

同じディレクトリに userfunc.c と *model*.ufn とが共に存在する場合は以下の優先順位となります。

＜リンクされるユーザ定義関数＞

順位	条件（ソースファイルの存在）	採用されるソースファイル
1	*model*.ufn と userfunc.c が共にある場合。	*model*.ufn
2	*model*.ufn が無く userfunc.c が有る場合	userfunc.c
3	*model*.ufn、 userfunc.c 共に無い場合	無し

6.6　　ＭＬ4.2c とＭＬ5.1 の共用

ユーザ定義関数を、ＭＡＴＬＡＢ の 4.2c と 5.1 の両方の版について、ソースファイルを共用させたい場合で、版の相違により、どうしても異なる記述が要求される場合があります。

この場合、ＤＳＰ用開発環境で定義されている定数を使って条件付きコンパイルをする事ができます。

参照する定数は　ML51 です。

この定数は、ＭＡＴＬＡＢ 4.2c の場合には未定義、ＭＡＴＬＡＢ 5.1 の場合には 値が１と定義されています。従って下記の様な使い方ができます。

```
#if       ML51
          /* MATLAB 5.1 用コードを記述 */
#else   /* of ML51 */
          /* MATLAB 4.2c 用コードを記述 */
#endif   /* of ML51 */
```

7　制限事項

7.1　　動作上の機能制限

ＭＡＴＬＡＢ対応ライブラリは、Ｉ／Ｏブロックが、実時間実行ではないものの、SIMULINK 画面から利用できるのが、ひとつの特徴です（ＭＡＴＬＡＢ 4.2c のみ)。
しかし、ユーザ関数組み込み機能については、SIMULINK 画面からは利用できません。
SIMULINK 画面でシミュレーションを実行した場合、弊社製品の有無にかかわらず、トップレベルのブロック線図の Inport については常に０が入力され、又、Outport については出力は無視されます。

ユーザ関数とのインターフェースを、SIMULINK 画面で確認するには、Inport の代わりにダミーのサブブロックを、Outport の代わりに Scope や To-Workspace 等を使い、模擬的に行なってください。

7.2　　使用出来ないソースファイル名

ユーザ定義関数では、利用出来るソースファイルの名称に制限があります。本製品で使用しているソースファイル名称と重複すると、不都合が生じますので注意してください。
本製品では、以下のソースファイルを使用しています。尚、版により若干変更がある場合がありますので、念の為、ディレクトリを直接参照し、確認する事をお勧めします。

7.2.1　　MATLAB 4.2c の場合

● ディレクトリ：MATLAB¥CODEGEN¥RT¥CHUBU¥OS

● ヘッダファイル

ADSP32X.H
ADSPTYPE.H
CTRL.H
HEAPEX.H
LOGOPT0.H
MINIMON.H
RT_CPU.H
RT_ERROR.H
RT_LOG.H
RT_SIM.H
RT_TRIG.H

● ソースファイル

RT_ACCES.C
RT_BKGND.C
RT_CPU.C
RT_EULER.C
RT_EVENT.C
RT_HUEN.C

RT_INTEG.C
RT_LOG.C
RT_MAIN.C
RT_MON.C
RT_NONE.C
RT_RK3.C
RT_RK5.C
RT_SIM.C
RT_SIMEV.C
RT_TIMER.C
RT_TRIG.C
USERFUNC.C

7.2.2 MATLAB 5.1 の場合

● ディレクトリ：MATLAB¥RTW¥C¥ADSP32X¥RT

● ヘッダファイル

ADSP32X.H
ADSPTYPE.H
CTRL.H
EXTRNFCN.H
HEAPEX.H
LOGOPT.H
LOGOPT0.H
MINIMON.H
RT5COMP.H
RT5CPU.H
RT5SIM.H
RT_ERROR.H
RT_LOG.H
RT_TRIG.H

●ソースファイル

ODE0.C
ODE1.C
ODE2.C
ODE3.C
ODE4.C
ODE5.C
RT5CPU.C
RT5EVENT.C
RT5GETT.C
RT5MAIN.C
RT5ODE.C
RT5SIM.C
RT5TIMER.C
RT_ACCES.C
RT_BKGND.C
RT_LOG.C
RT_MON.C
RT_TRIG.C
USERFUNC.C
及び *MODEL*.UFN

7.3　　長時間に渡るＤＳＰの専有

バックグランド処理はアイドルループを回りながら、ＲＴＭＯＮからのアクセスに応答する傍ら実行されます。ＲＴＭＯＮからのアクセスとは、例えばゲイン値変更とか、変数表示等をさしています。
従って、極端な例では、ユーザ定義のバックグランド処理で無限ループを構成すると、ＲＴＭＯＮからのアクセスに応答が出来ず、ＲＴＭＯＮからはＤＳＰが止まったか、に見えてしまいます。

無限ループを構成しないまでも、出来るだけ短い時間で、一旦関数から戻る様な記述をしてください。

例として、ユーザ定義関数のバックグランド処理で、ある条件が整うのを待機する様な処理を記述したい場合、以下の様に、条件が整っていなければ一旦関数から戻り、次回呼び出された時再度継続する様な記述としてください。

バックグランド処理記述の例

```
static int    flg=0;
void UserBackground( )
{
        switch( flg ){
                case 0:  if( ! ConditionOk( ) )        /*  条件が整わなければ  */
                                return;                /*  一旦戻る  */
                         else    flg = 1;              /*  整ったらフラグをセットし  */
                         return;                       /*  一旦戻る  */
                case 1:  NextProc();                   /*  条件が整った後の処理  */
                         flg = 2;                      /*  フラグセットし  */
                         return;                       /*  戻る  */
                default: return;                       /*  以後何もしない  */
        }
}
```

この例の想定は、関数 ConditionOK( ) は条件が整っているか確認する関数で、条件は長時間の渡り整わない可能性があるものとします。この関数は、条件が整ったら TRUE を、それ以外は FALSE を返すものとします。

7.4　　長時間に渡る割り込み禁止

ＲＴモデルでのブロック線図の計算は、タイマー（又は外部からの）割り込みにて計算を駆動しています。従って、ST レジスターの GIE の操作により、割り込みを禁止するとブロック線図の計算が停止してしまいます。

出来れば、IE レジスターの停止したい割り込みに相当するビットの操作にて割り込みを抑止してください。

どうしても ST レジスターの GIE の操作により割り込みを禁止する必要がある場合は、出来るだけ短時間に抑えてください。禁止して良い時間の目安は、ステップサイズー実行時間（ブロック線図計算に要する時間）以内です。

ユーザ定義関数

第４版　2000.06.28

中部電機株式会社 ＣＳ事業部
TEL　0532-61-9566
FAX　0532-63-1081

ＭＡＴＬＡＢ対応ライブラリ（マルチ）
ADSP324-41B／ADSP404-241B／ADSP674-341
リアルタイムモニター・チュートリアル

中部電機株式会社

＜＜　目　　次　＞＞

1　概要 3
2　用語・略語の説明 3
3　モデルの作成 4
3.1　ユーザプログラミングによるモデル 4
3.2　ＳＩMULＩNKによるモデル 4
3.2.1　作業用フォルダーの作成 4
3.2.2　カレントディレクトリの設定 4
3.2.3　モデルの作成 6
3.2.4　モデルの評価 12
3.2.5　コード化 14
4　ＲＴMONの起動 17
4.1　ＲＴMONの起動 17
4.2　パスワードの入力 17
5　モデルペアーの作成 18
5.1　モデルペアーの概念 18
5.2　モデルペアーの新規作成 18
5.3　モデルペアーの編集 23
6　モデルのロード 24
6.1　シングルＤＳＰモデルのロード 24
6.2　マルチＤＳＰモデルのロード 25
7　実行パラメータ（ＲＴオプション）変更 26
8　実行開始 29
9　変数モニター 30
9.1　変数の選択 30
9.2　ベクトル変数の選択 31
9.3　表示点数の設定 32
9.4　スコープの自動配置 33
9.5　変数の数値読み取り 34
9.6　スコープのレンジ変更 34
9.7　Ｙ軸オフセットを付ける 35
9.8　Ｙ軸オフセットの除去 36
9.9　変数の削除 36
9.10　スコープ状態の保存・選択 36
10　計算所要時間の測定 37

11 データロギング 39
11.1 ロギング項目 39
11.2 ロギング項目選択 40
11.3 データ間引き 40
11.4 データ点数 41
11.5 ファイル形式・ファイル名 41
11.6 マニュアルトリガー 41
12 モデルの終了 42
13 ロギングデータの保存 43
13.1 シングルモデルの場合 43
13.1.1 上書き保存 43
13.1.2 新規保存 43
13.2 マルチモデルの場合 43
14 ロギングデータのＭＬでの利用 45
15 自動テスト用ブック作成 46
15.1 基本ブック指定 46
15.2 作成ブック指定 47

1　概要

本書は、ＭＡＴＬＡＢ、ＳＩＭＵＬＩＮＫ（共に米国 The Math Works Inc 社製品）及びＭＡＴＬＡＢ対応ライブラリマルチ「ＡＤＳＰ３２４－４１Ｂ」「ＡＤＳＰ４０４－２４１Ｂ」「ＡＤＳＰ６７４－３４１」（中部電機（株）製品）を用いて、実時間アプリケーションを作成する具体的手順を、実例を元に説明しています。
本書で説明している各製品のバージョンは以下の通りです。

ＭＡＴＬＡＢ等	5.1.1、5.2、5.3、5.3.1 版
ＭＡＴＬＡＢ対応ライブラリ（マルチ）	4.01 版
ＤＳＰ用コンパイラ（C3x）	5.0、5.1 版
ＤＳＰ用コンパイラ（C6x）	1.1 版

2　用語・略語の説明

本書では説明の簡略化の為、次の用語（略語）を使用する場合があります。

ディレクトリ	フォルダーと同じ意味
ＤＬＧ	ダイアログボックス
ＭＬ	ＭＡＴＬＡＢ
ＭＬ－ＬＩＢ	MATLAB 対応ライブラリ
ＲＴＷ	Real-Time Workshop
ＲＴＭＯＮ	リアルタイムモニター（本製品に含まれる）
ＲＴモデル	リアルタイムモデル
ＳＬ	ＳＩＭＵＬＩＮＫ
ＴＩ	Texas Instruments （DSP チップ製造元）
ＴＭＷ	The Math Works Inc. （MATLAB 製造元）

3　モデルの作成

本製品で取り扱える実時間モデルの作成には、２つの方法があります。

3.1　ユーザプログラミングによるモデル

１つは目、汎用エディターでＣ言語又はアセンブラを用いてプログラミングし、弊社ソースレベルデバッガ／ＭＳＣアダプタ等の製品又は、ＴＩ社製エミュレータの何れかを用いてデバッグを行い、実行可能形式にしたものです。本製品では、この形式の実時間モデルについては、ロード・実行はできますが、メモリー内部の値や計算結果のモニター等を行う事は出来ません。デバッガ等を用いて完全にデバッグを行ったうえ、ロードすればただちに実行できる形式にしてください。

尚、本書ではこの方法については説明を行っていません。

3.2　ＳＩＭＵＬＩＮＫによるモデル

2 つめは、ＳＬにより作成したブロック線図を元にＲＴＷを用いてＣ言語ソースを経て実行形式プログラムを自動生成する方法です。以下本書ではこれをＲＴモデルと呼びます。例として、ＤＳＰで正弦波を生成し、これをＤＡ変換器に出力すると共に、ＡＤ変換器からデータを取り込むモデルを作ってみます。

3.2.1　作業用フォルダーの作成

ＭＬにはカレントディレクトリという概念があり、作成されたブロック線図や、ＲＴＷで作成するＲＴモデルは全て、このカレントディレクトリに保存されます。この作業用フォルダは、エクスプローラ等のファイル管理ツールで作成します。

3.2.2　カレントディレクトリの設定

次に、ＭＬを起動します。

ＭＬ起動後、最初に行う作業は、カレントディレクトリを、希望とするフォルダーに移動する事です。

ＭＬインストール後、通常に起動すると、カレントディレクトリは C:¥Matlab¥Bin になっています。MATLAB Command Window に ｃｄ と入力すると現在のカレントディレクトリが表示されます。

＜例＞

》 cd

C:¥Matlab¥Bin

これを例えば、D:¥USER¥SomeOne¥ml51test に移動するには、
》cd　D:¥USER¥SomeOne¥ml51test
と入力します。

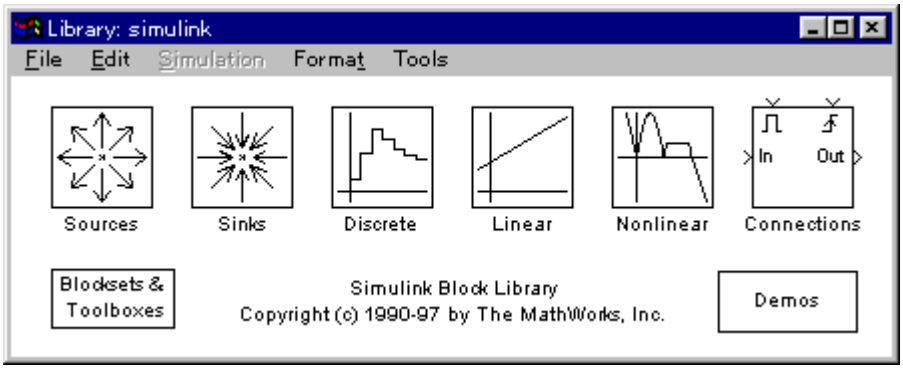
もし、毎回入力せず、自動的にカレントディレクトリを移動させたいのであれば、例えばデスクトップにショートカットを作成し、このプロパティーのショートカットの作業フォルダー(S)にカレントディレクトリを入力しておきます。

＜ショートカットの設定例＞

ＭＬ起動時のカレントディレクトリを自動的に移動する方法は他にもありますので、どの方法を用いても構いませんが、要点は、保存したブロック線図やデータファイル等を見失わないで済む為、これからＭＬやＳＬで作業をしようとする際には、必ず「カレントディレクトリが何処になっているか？」を確認する事です。

3.2.3　モデルの作成

続いて MATLAB Command window に simulink と入力し、ＳＬを起動します。
次の要な、ＳＬの標準ブロックライブラリと、中身が空のブロック線図が開きます (注)。



＜ＳＬ標準ブロックライブラリ＞

中身が空のブロック線図はタイトルが untitled となっています。これは、まだユーザがこのブロック線図に名称を付けていないので、暫定的にＳＬが付けたものですので、ブロック線図メニューから、File(F) ～ Save As(A)で名称を付けて保存します。例えば test 等とします。拡張子は省略すれば、自動的に .mdl が追加されます。この時、フォルダーはＭＬカレントディレクトリになっていますので、変更しないようにします。

ファイル名称は、ＲＴＷで使用する場合、制約がありますので次の制約を守って命名してください。

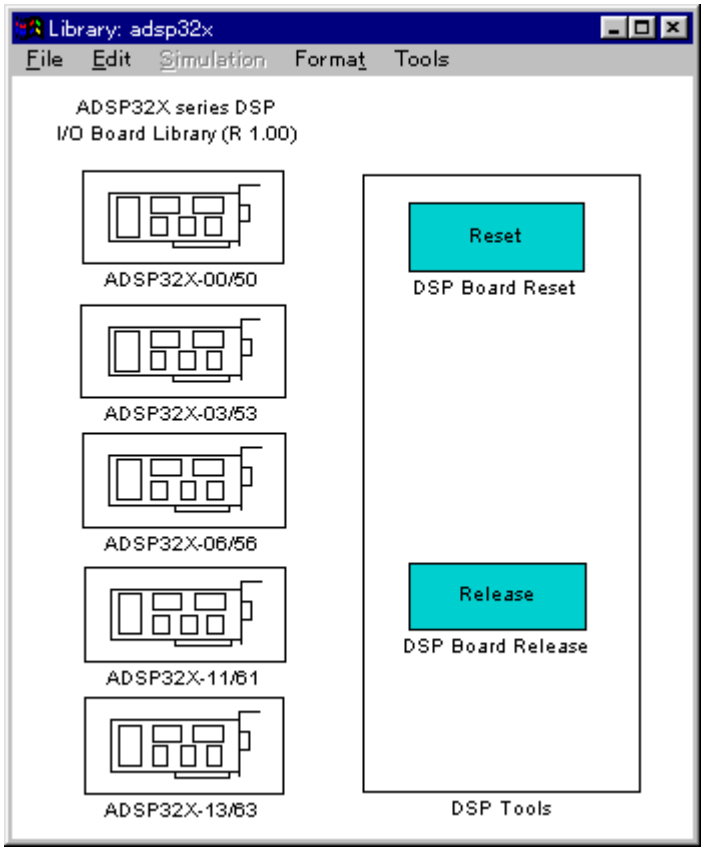
（１）　ファイル名称は８文字以内とします。
　　（例：abcdefgh.mdl＝ＯＫ、abcdefghi.mdl＝ＮＧ）
（２）　数字で始まってはいけません。__ 又は アルファベットで始めます。
　　（例：_model.mdl、model.mdl＝OK、1model.mdl＝ＮＧ）
（３）　かな、漢字は使用してはいけません。
　　（例：abc_123.mdl＝ＯＫ、ｶﾅ.mdl、漢字.mdl＝ＮＧ）
（４）　__以外の記号は使ってはいけません。
　　（例: abc_123.mdl＝ＯＫ、abc$.mdl、abc%&.mdl＝ＮＧ）
（５）　大文字小文字は、同じと見なされます。

こうして保存したブロック線図は、次回ＭＬ起動時に、MATLAB Command Windw にブロック線図の名称を入力する事により、開く事ができます。

注：MATLAB Command Window から直接ブロック線図の名称を指定する等して、既にブロック線図が開いている場合は、新しいブロック線図は自動的には作成されません。その場合は、ＳＬ標準ラ

イブラリのメニューから、 File ～ New ～ Model でブロック線図を開いてください。

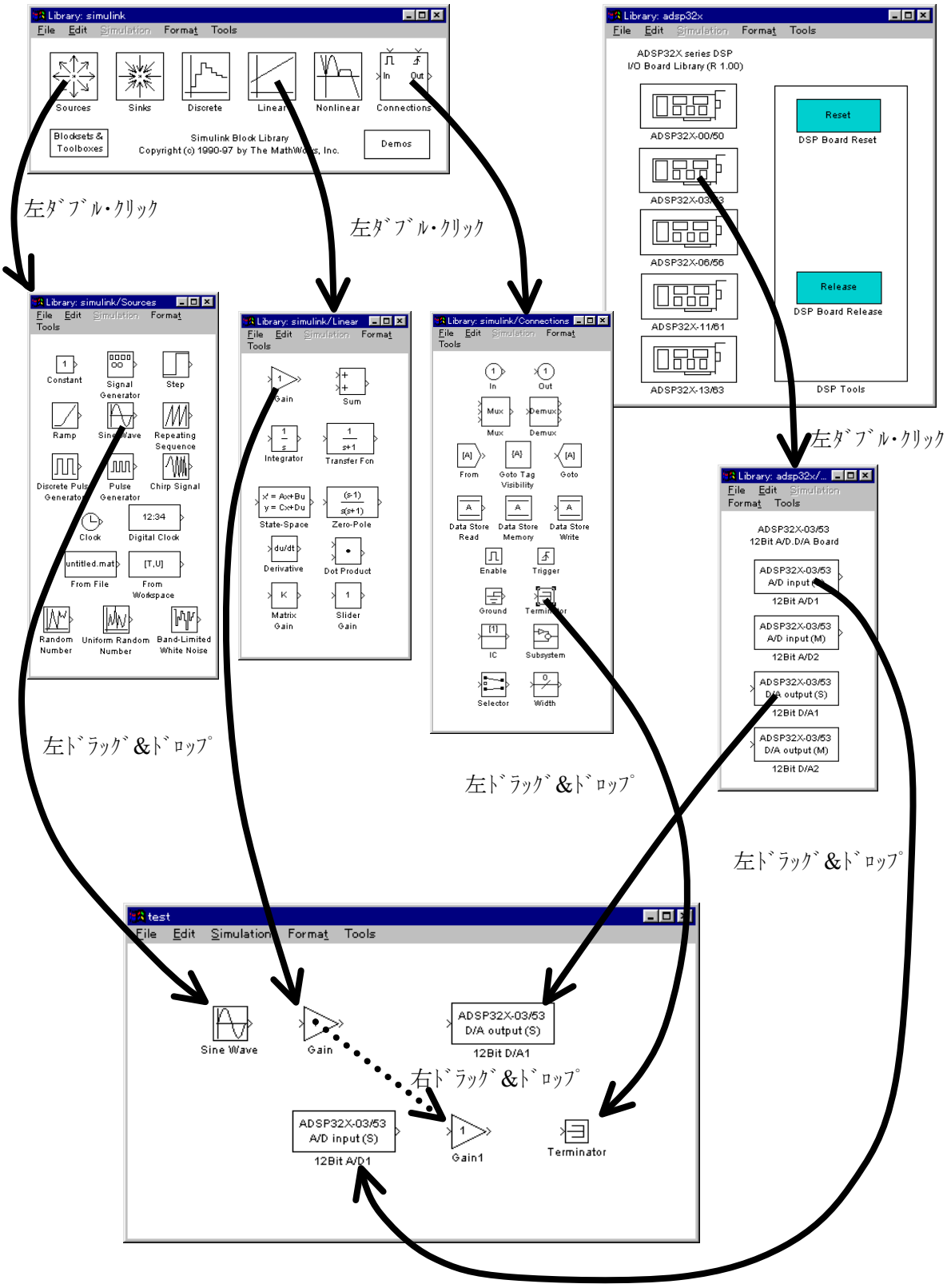
次に、ＭＬ－ＬＩＢのライブラリを開く為、MATLAB Command Window に対して、adsp32x と入力します。次のウィンドウが開きます。



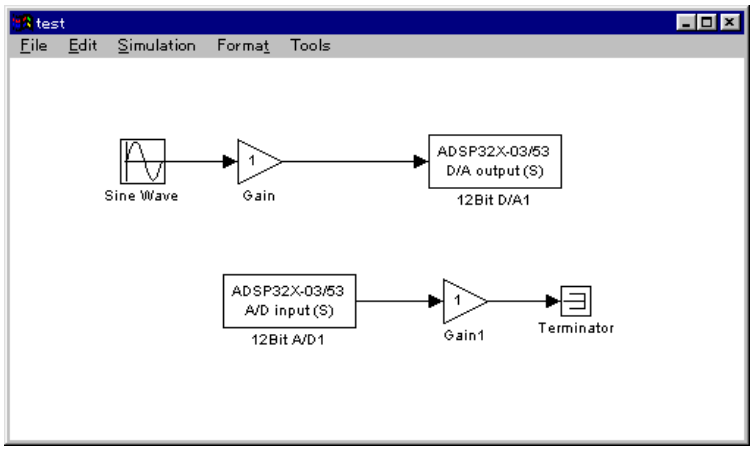
＜ＭＬ－ＬＩＢ ブロックライブラリ＞

これらブロックライブラリの中から必要なライブラリをドラッグし、ブロック線図にドロップします。

正弦波は、ＳＬブロックライブラリの中の Sources に入っていますので、まず Sources をマウス左ボタンでダブルクリックし、開いたライブラリの中から Sine Wave をマウス左ボタンでドラッグし、ブロック線図にドロップします。以下同様に次ページに示すブロックをドラッグ＆ドロップしてください。


Library: simulink
File Edit Simulation Format Tools
Sources
Sinks
Discrete
Linear
Nonlinear
Connections
Blocksets & Toolboxes
Simulink Block Library
Copyright (c) 1990-97 by The MathWorks, Inc.
Demos
左ﾀﾞﾌﾞﾙ･ｸﾘｯｸ
Library: adsp32x
ADSP32X series DSP
I/O Board Library (R 1.00)
ADSP32X-00/50
ADSP32X-03/53
ADSP32X-06/56
ADSP32X-11/61
ADSP32X-13/63
Reset
DSP Board Reset
Release
DSP Board Release
DSP Tools
左ﾀﾞﾌﾞﾙ･ｸﾘｯｸ
Library: simulink/Sources
Constant
Signal Generator
Step
Ramp
Sine Wave
Repeating Sequence
Discrete Pulse Generator
Pulse Generator
Chirp Signal
Clock
Digital Clock
From File
From Workspace
Random Number
Uniform Random Number
Band-Limited White Noise
左ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟ
Library: simulink/Linear
Gain
Sum
Integrator
Transfer Fcn
State-Space
Zero-Pole
Derivative
Dot Product
Matrix Gain
Slider Gain
Library: simulink/Connections
In
Out
Mux
Demux
From
Goto Tag Visibility
Goto
Data Store Read
Data Store Memory
Data Store Write
Enable
Trigger
Ground
Terminator
IC
Subsystem
Selector
Width
左ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟ
Library: adsp32x/...
ADSP32X-03/53
12Bit A/D.D/A Board
ADSP32X-03/53 A/D input (S)
12Bit A/D1
ADSP32X-03/53 A/D input (M)
12Bit A/D2
ADSP32X-03/53 D/A output (S)
12Bit D/A1
ADSP32X-03/53 D/A output (M)
12Bit D/A2
左ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟ
test
File Edit Simulation Format Tools
Sine Wave
Gain
ADSP32X-03/53 D/A output (S)
12Bit D/A1
右ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟ
ADSP32X-03/53 A/D input (S)
12Bit A/D1
Gain1
Terminator

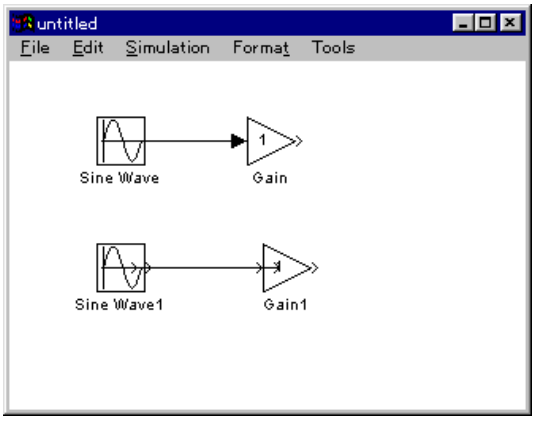
次に、ブロック線図に貼り付けたブロックの出力から次のブロックへの入力へと、マウス右ボタンでドラッグし結線します。



＜結線が終わったブロック線図の例＞

結線が正しく接続出来ている線は ⟶ の形になっています。
次の例は、上側のブロック（Sine Wave と Gain の間）は入力出力共に正しく結線出来ていますが、下側（Sine Wave1 と Gain1 との間）は入力出力共に正しく結線出来ていません。ブロックと線の端が重なっているので、一見つながっているかに見えますが、線の形でつながっていない事が判ります。

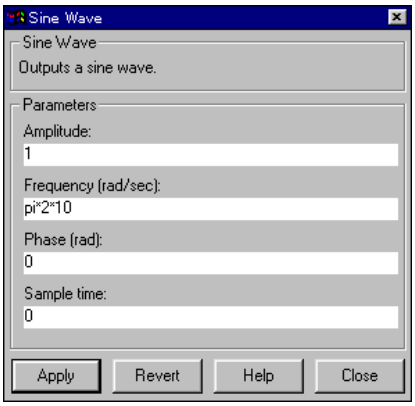
＜結線不良（下側）の例＞



この場合は線を一旦削除して引き直すか、又はブロックアイコンをマウス左でドラッグして一旦位置を移動し、正しく結線出来た後元の位置に戻します。

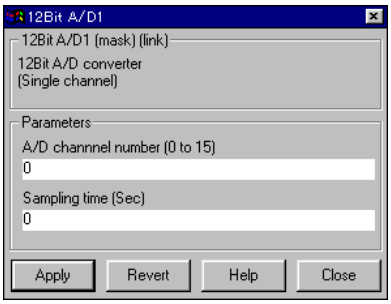
結線が出来たら、パラメータを設定します。ここでは正弦波の周波数と A/D ブロックのサンプルタイムを変更します。
Sine Wave ブロックアイコンをダブルクリックすると下の初期設定ＤＬＧだ開きますので、Frequency［単位はラジアン／秒］を設定します。例えば１０Ｈｚであれば、pi＊2＊１０を設定し、Close ボタンを押します。

＜Sine Wave 初期設定ＤＬＧの例＞



Ａ／Ｄ変換器の初期設定ＤＬＧも同様に開き、下記の様にサンプルタイムを０に設定します。サンプルタイム０は連続時間系を意味し、ステップサイズ毎にＡＤからデータを取る事を意味します。

＜ＡＤ変換器の初期設定＞



尚、ＡＤとＤＡはこの例では共にＣＨ０を使います。これ以外のＣＨの場合や、複数のＣＨを使う（複数のブロックを使う）場合は適宜ＣＨ番号の設定が必要です。

設定が出来たら、ブロック線図のメニューから、File ～ Save でブロック線図を保存します。

3.2.4　モデルの評価

モデルが希望通り動作するか否かは、ＳＬ上で評価出来る場合と出来ない場合があります。例えば外部機器（制御対象等）と接続しないといけない等、ＳＬだけでは評価出来ないのが一般的ですが、出来れば擬似データ等を用いて予め、ある程度の目処を付けておく事をお勧めします。

その際に注意が必要なのは、**評価の最終段階では、積分アルゴリズムを「固定ステップサイズ」にしても大丈夫か確認する事**です。

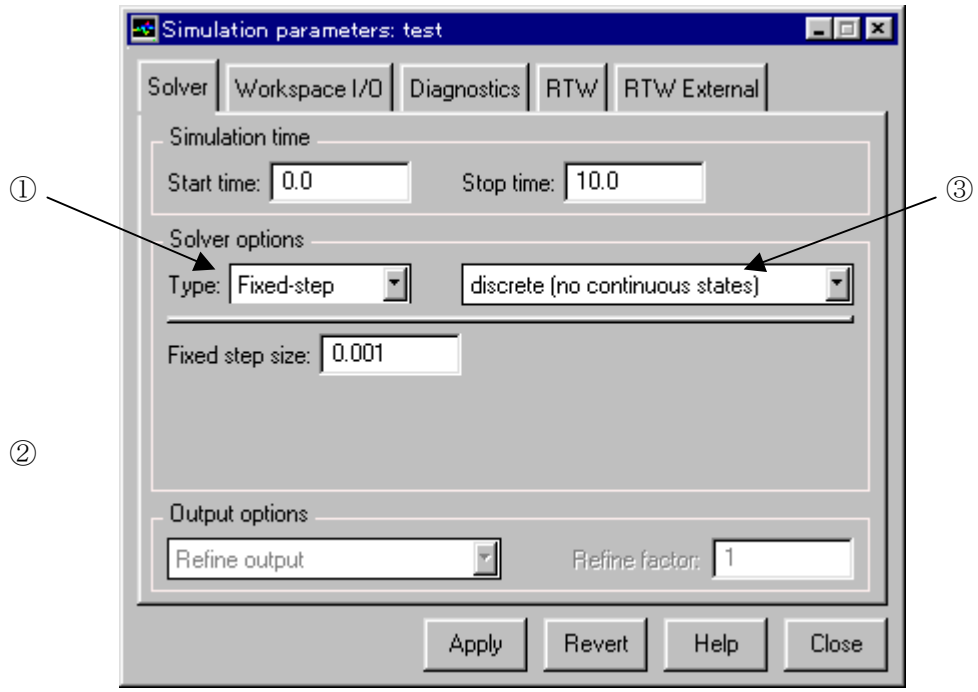
一般に数値シミュレーションでは、計算精度と計算時間の短縮の為、可変時間刻みのアルゴリズムが良く用いられます。これは、状態変数の導関数を計算し、傾きが急な場合計算精度を上げるため時間刻みを短くし、逆に傾きが緩やかな場合は時間刻みを長くしても計算精度は低下しないので、計算回数を減らして計算に費やす時間を節約する（計算回数を減らす事は同時に誤差の累積を減らす意味をも含む）為、時間刻みを長くします。ＳＬでもこれら、可変時間刻みのアルゴリズムが多数組み込まれています。

しかし、最終目的が「実時間モデルを作る事」の場合、実時間モデルでは可変時間刻みが実現出来ない為、ＳＬで評価の段階で、固定時間刻みで評価をしておく必要があります。固定時間刻みの場合、必然的に時間刻みの幅（ステップサイズ）により計算結果が違って来る場合がありますので、どの程度のステップサイズで計算すれば良いかも確認しておく必要があります。

固定時間刻みにするには、ブロック線図のメニューから Simulation ～ Parameters..を選択し次の Simulation parametersＤＬＧを開きます。このＤＬＧの Solver タブを選択し、Solver options グループの Type コンボボックスから Fixed- step を選択します。続いてすぐ下の Fixed step size:に希望するステップサイズを入力します。auto のままではＲＴＷでコード化出来ません。最終的には必ず、ある具体的なステップサイズ値を入力してください。

又、モデルに連続系の状態変数が無いのに、discrete 以外の積分アルゴリズムを選択した場合は、警告が出ます。コード生成の時も同様に警告されますので、この時点で discrete に変更してください。

次の例は、固定時間刻みで積分アルゴリズム無し（今回使う例 test.mdl では積分アルゴリズムが不要）に設定した例です。



＜シミュレーションパラメータ設定例＞

①ここで固定／可変時間刻みを選択します。Fixed-step が固定時間刻みです。
②ステップサイズを設定します。単位は秒です。具体的な数値を入力してください。
③積分アルゴリズムを選択します。
まず最初に、①で固定／可変を選択します。これによりＤＬＧの表示項目が変わりますので、続いて②と③を設定します。

3.2.5　コード化

評価が終わったモデルは次の手順でコード化します。

ブロック線図のメニューから、Tools　～ RTW Options..　を選択し、Simulation parametersＤＬＧを開きます。

Solver タブを選択し、Solver optins グループの Type が Fixed-step である事、その右の積分アルゴリズムが希望するアルゴリズムである事、その下の Fixed step size:も同様に希望するステップサイズ値になっている事を確認します。

この値は、ＤＳＰで実行する際の設定とは直接には関係しません（ＲＴＭＯＮではブロック線図の設定とは別に、自由に設定できます）が、ＲＴＭＯＮでＲＴモデルをロードした際にデフォルト設定として呼び出す事ができます。

次に RTW タブを選択し、以下を設定してください。

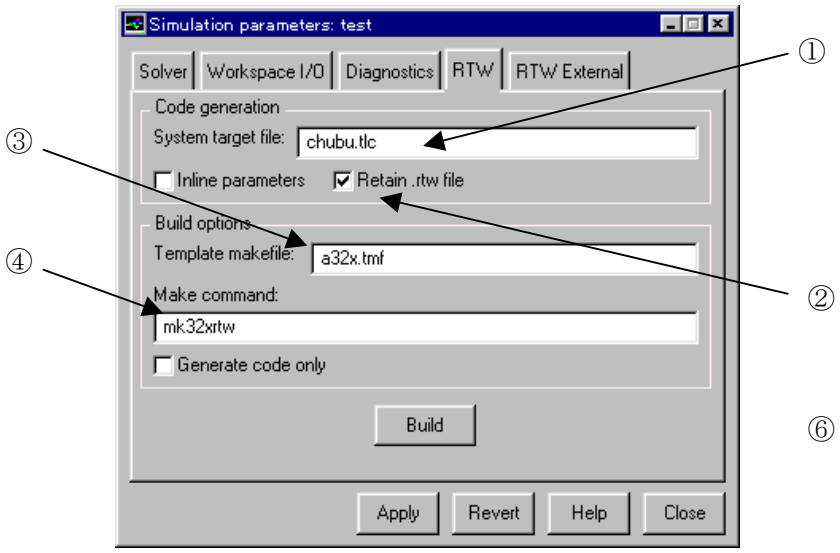
①System target file: chubu.tlc

②Retain .rtw file（チェック状態にします）

③Template makefile: a32x.tmf　(ADSP404:a404.tmf, ADSP674:a674.tmf or a674b.tmf)

④Make command: mk32xrtw(ADSP404:mk404rtw,ADSP674:mk674rtw or mk674brtw)

⑤Generate code only（チェック無しの状態にします。通常はこうなっています。）



＜ＲＴW設定ＤＬＧ設定例＞

この状態で一旦ブロック線図を保存します。そうすれば、万が一ＰＣがハングした場合でも、行った設定はディスクに残っており、次回ＭＬ起動時にブロック線図を開けば設定が復元されます。

注）新規以外で作成されたモデルをコード化する場合、弊社ホームページの Q&A「ＭＡＴ

⑤

ＬＡＢ対応ライブラリでよくあるＱ＆Ａ」を参照してください。

コード生成の実施は、このＤＬＧのＲＴＷタブを選択し、Build ボタン（図の⑥）を押します。
MATLAB command window に以下の表示がなされ、コード生成が完了します。
本製品の版番号や、生成するＲＴモデルの実行形式、及び、初めてコード生成するか２度め以後かにより、表示される行数や内容が異なります。
良否の判断は、全ての行が ### で始まっており（コード生成時のエラーメッセージは### で始まらない）、かつ、最後の 2 行が以下と同様（但し test の部分は個々のブロック線図の名称）であれば、良好と判断してください。

```
### Starting RTW build procedure for model: test
### Invoking Target Language Compiler on test.rtw
### Loading TMW TLC function libraries
### Initial pass through model to cache user defined code
### Creating source file test.c
### Creating part 1 of registration file test.reg
### Creating parameter file test.prm
### Creating model header file test.h
### Creating part 2 of registration file test.reg
### TLC code generation complete.
### test.mk which is generated from D:¥MATLAB.511¥rtw¥c¥adsp32x¥a32x.tmf is up to date
### Building test: nmake /nologo -f test.mk STEP=0.001
### Compiling test.c
### Compiling rt5sim.c
### Compiling rt5main.c
### Compiling rt_log.c
### Compiling rt_trig.c
### Compiling rt_acces.c
### Compiling rt_mon.c
### Compiling rt_bkgnd.c
### Compiling rt5gett.c
### Compiling rt5cpu.c
### Compiling rt5timer.c
### Compiling rt5ode.c
### Compiling ode0.c
### Compiling ode1.c
### Compiling ode2.c
### Compiling ode3.c
### Compiling ode4.c
### Compiling ode5.c
### Linking test.out
### Created
### Successful completion of RTW build procedure for model: test
```

生成されたＲＴモデルの実行ファイルは拡張子.OUT（例：*model*.out）で、ＭＬのカレントディレクトリに格納されます。

4　ＲＴＭＯＮの起動

4.1　ＲＴＭＯＮの起動

ＲＴＭＯＮは、Ｗｉｎｄｏｗｓのスタートメニューから スタート ～ プログラム(P) ～ ﾘｱﾙﾀｲﾑﾓﾆﾀ　～ ﾘｱﾙﾀｲﾑﾓﾆﾀｰ（ﾏﾙﾁ）を選択して起動します。

4.2　パスワードの入力

購入後又は再インストール後、初めての起動の場合、パスワードの入力を要求されます。製品購入時に添付されていたライセンス・シートに記載されているパスワードを入力してください。

パスワードは、１６桁の英数字です。英字は大文字小文字は区別されます。数字の０（ゼロ）と大文字のＯ（オー）等は間違え易いので、ライセンス・シート記載されているパスワードの下の記号列（ＮＮＡＡ等）を参照の上正しく入力してください。

＜パスワード入力の例＞

5　モデルペアーの作成

5.1　モデルペアーの概念

これからロードしようとするＲＴモデルがマルチＤＳＰの場合、各ＤＳＰにＲＴモデルを一括してロード出来る様にする為に必要な情報である、何台のＤＳＰを使うか？、どのＲＴモデルを何番のＤＳＰにロードするか？等を定義し、これらの情報をＲＴモデルとは別に一括して保存・管理します。これを「モデルペアー」と呼びます。ＤＳＰを１台しか使わない場合は必要ありません。

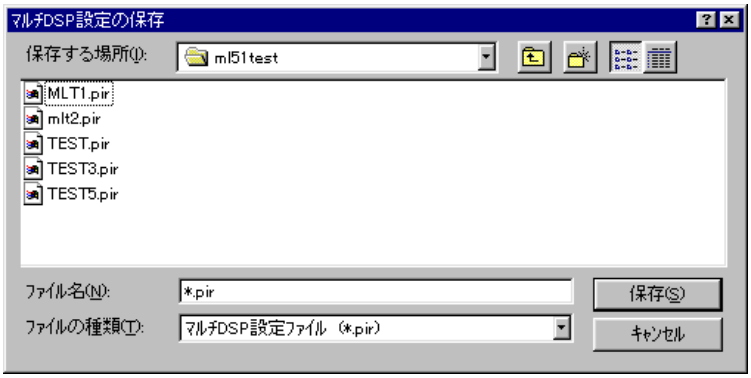
マルチＤＳＰの場合、各ＤＳＰにはそれぞれ１つづつＲＴモデルを割り当てます。ＲＴモデルはブロック線図によるものでも、Ｃ言語やアセンブラで記述したものでも構いません。又、同じＲＴモデルを複数のＤＳＰに割り当てても構いません。

5.2　モデルペアーの新規作成

ここでは、０、１、２番の計３台のＤＳＰを使い、各ＤＳＰにそれぞれ、dsp0.out、dsp1.out、dsp2.out をロードする場合を想定します。

まず、ＲＴＭＯＮのメニューから、 ファイル(F) ～ 新規ﾏﾙﾁ DSP 設定(N)を選択し、マルチＤＳＰ設定の保存ＤＬＧを開きます。

＜マルチＤＳＰ設定の保存ＤＬＧ＞

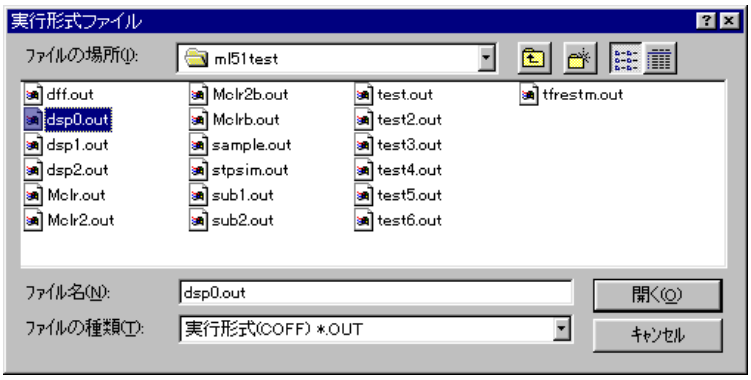


こうして保存されたモデルペアーは拡張子 .PIR ファイルとして保存されます。

ファイル名称を決定しますと、マルチＤＳＰ設定ＤＬＧが開きます。

＜マルチＤＳＰ設定ＤＬＧ＞

開いた直後のＤＬＧは全て空になっています。ＤＳＰボード番号の表示が０になっており０番のＤＳＰへロードするモデルが未だ設定されていない事を示しています。
最初に０番ＤＳＰの設定をします。メインモジュールの右の参照ボタン（２個あるうちの上側）を押し、実行ファイル選択ＤＬＧを開きます。このときＤＬＧ左下にある「相対パス指定」にチェックを入れておく事により、モデルペアーファイルを基準としたＲＴモデルの相対位置関係を崩さない限り他のディレクトリに移動させてもモデルのロードが可能になります。



＜実行ファイル選択ＤＬＧ＞

ロードしたいＲＴモデル、この場合 dsp0,out を選択します（上の図）。ＲＴモデルはＭＬのカレントディレクトリに作成されていますので、このＤＬＧでは“ファイルの場所”を使ってフォルダーを間違えない様にしてください。ＭＬとＲＴＭＯＮは直結していませんので、ＭＬでカレントディレクトリをセットしてあっても、その設定はＲＴＭＯＮには直接反映していません。

選択したら、開く(O)ボタンを押します。すると、マルチＤＳＰ設定ＤＬＧに今選択したＲＴモデルのディレクトリ名とファイル名の拡張子を除いた部分が表示されます。（下図）これで０番ＤＳＰへロードするＲＴモデルの設定は完了です。補助モジュールの欄は、通常は空白のままで構いません。

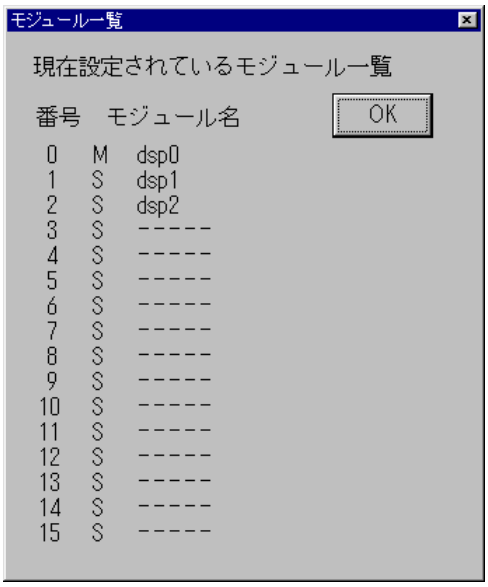
＜マルチＤＳＰ設定ＤＬＧ＞

次に、１番ＤＳＰにロードするＲＴモデルを設定する為、ＤＳＰボード番号を切り替えます。右上のスライダー①をマウス左ボタンで少し右にドラッグするか、又はスライダーの右端にあるボタン②を１回、マウス左ボタンで押します。ＤＬＧの表示は次の様に変わります。

ボード番号の表示が１に変わり、１番ＤＳＰの設定が未だ完了していない事を表しています。先の０番ＤＳＰの場合同様に参照(R)ボタンによりＲＴモデルを選択します。

同様にして０～２番ＤＳＰまで全て設定したら、一覧（L）ボタンを押してＲＴモデルの一覧を表示し、希望通り設定されているか確認してください。

＜モジュール一覧表示ＤＬＧ＞



もしロードの不要なＤＳＰ番号に誤って設定してしまったら、スライダーでその番号のＤＳＰを選択し、削除ボタンで削除します。

次に、モデルをロードする順番を指定する為、順番（O）ボタンを押します。

＜マルチＤＳＰロード順番ＤＬＧ＞

（未設定）

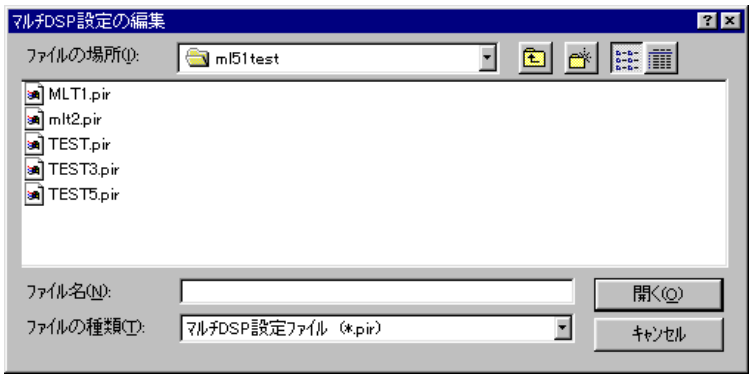
（設定完了）

未設定の状態では上図左の様に０番ＤＳＰしか設定されていません。
ＤＳＰ番号で、０→１→２の順にロードしたい場合は、Clear ボタン→０ボタン→１ボタン→２ボタンの順に押します。全てのＲＴモデルがブロック線図による通常のＲＴＭＯＮ

用モデルの場合は、若い番号順で良いでしょう。
番号を途中で間違えたら０番から正しい順で再度押して下さい。
上図右、設定完了の様に正しく設定出来たら、マルチＤＳＰロード順番ＤＬＧのＯＫボタンを押します。

5.3　モデルペアーの編集

既に作成してあるモデルペアーのファイルの内容を変更する場合は、ＲＴＭＯＮのメニューから ファイル(F) ～ 既存マルチＤＳＰ編集(E) を選択します。
マルチＤＳＰ設定の編集ＤＬＧが開きますので、ここで編集したいモデルペアのファイルを選択します。



＜マルチＤＳＰ設定の編集ＤＬＧ＞

モデルペアのファイルを選択すると、マルチＤＳＰ設定ＤＬＧが開き、前回保存した際の設定が表示されます。必要に応じて変更を加え、ＯＫボタンで保存してください。

＜マルチＤＳＰ設定ＤＬＧ＞

6　モデルのロード

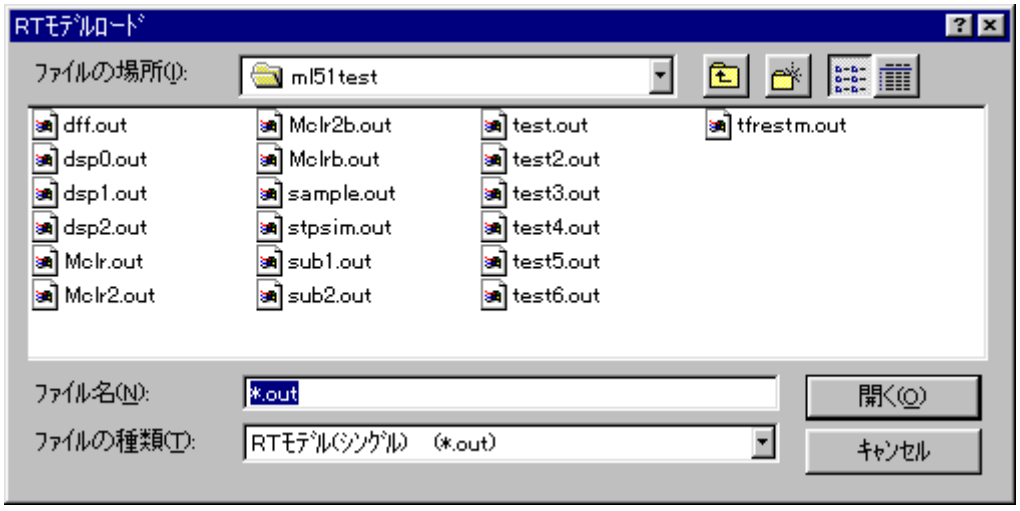
6.1　シングルＤＳＰモデルのロード

ＲＴモデルのロードはＲＴMONのメニューから、ﾌｧｲﾙ(F) ～ RT ﾓﾃﾞﾙﾛｰﾄﾞ(O) を選択します。

ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧが開きますので、ｼﾝｸﾞﾙ DSP の R ﾓﾃﾞﾙを選択し、更にﾛｰﾄﾞしたい DSP ﾎﾞｰﾄﾞ番号を設定（エディットボックスをマウス左ボタンでクリックし、キーボードで DEL キー、BS キー、数字キー尚等を使ってボード番号を直接設定）してください。（下図）続いてＯＫボタンを押してください。

ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければＯＫボタンだけで結構です。

RT ﾓﾃﾞﾙﾛｰﾄﾞＤＬＧが開きますので、ロードしたいＲＴモデルのファイル名を選択し、開くボタンを押してください。



これによりＲＴモデルがロードされ、モデルウィンドウが開きます。

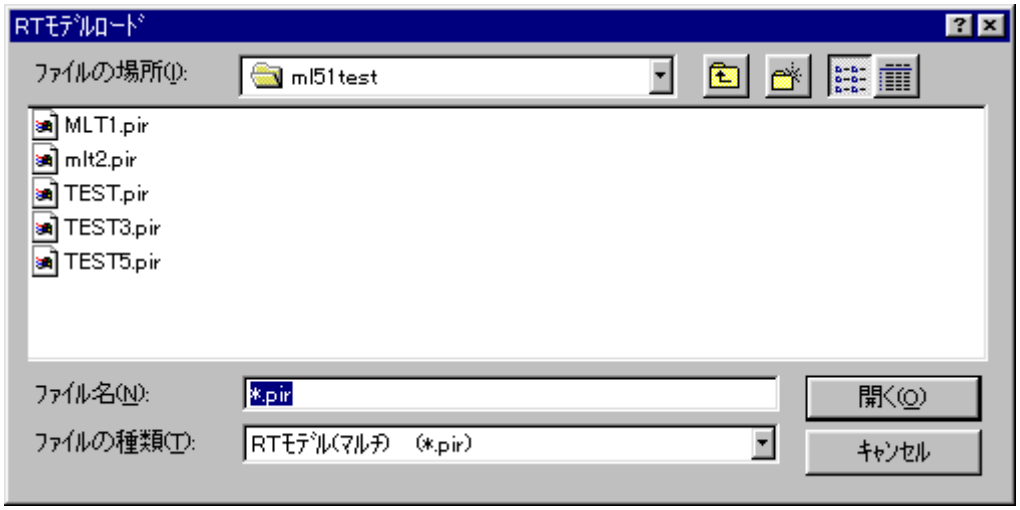
6.2　マルチＤＳＰモデルのロード

ＲＴモデルのロードはＲＴＭＯＮのメニューから、ﾌｧｲﾙ(F) ～ RT ﾓﾃﾞﾙﾛｰﾄﾞ(O) を選択します。

ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧが開きますので、ﾏﾙﾁ DSP の R ﾓﾃﾞﾙを選択します。（下図）続いてＯＫボタンを押してください。

ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければＯＫボタンだけで結構です。

RT ﾓﾃﾞﾙﾛｰﾄﾞＤＬＧが開きますので、ロードしたいモデルペアーのファイル名を選択し、開くボタンを押してください。

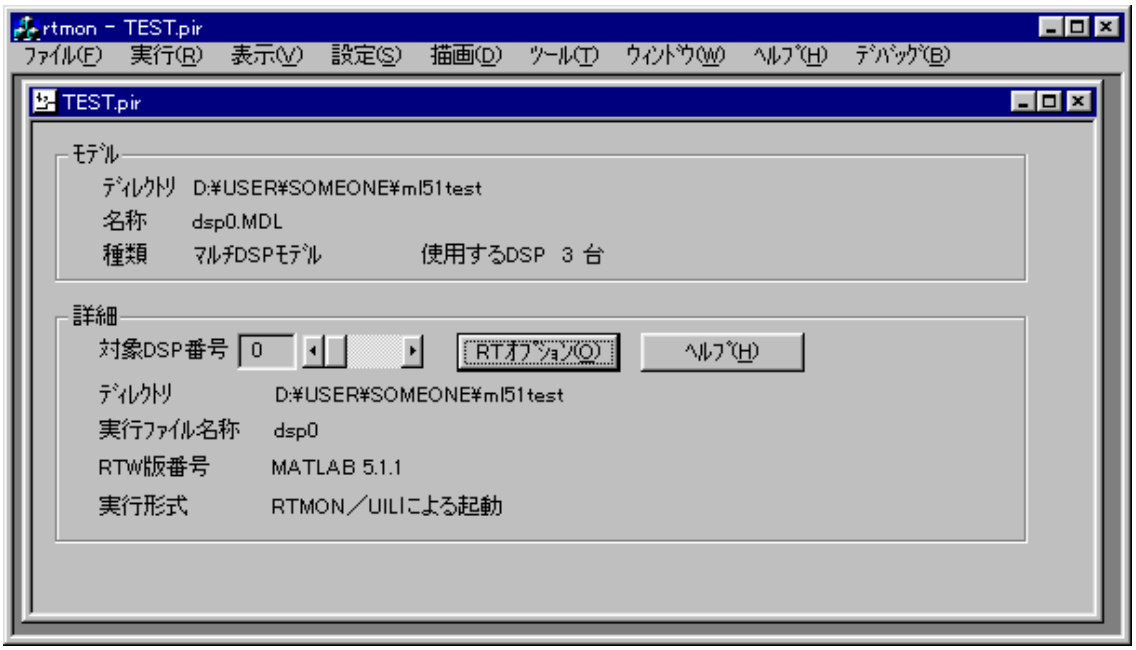


これによりＲＴモデルがロードされ、モデルウィンドウが開きます。

7　実行パラメータ（ＲＴオプション）変更

積分アルゴリズム、ステップサイズ等、ＲＴモデル計算の実行パラメータは、実行に先立ちＲＴＭＯＮから決定します。ＲＴＭＯＮでの設定はブロック線図の設定を反映させる事もできますが、それとは別にＲＴＭＯＮ上で一時的に異なるパラメータで実行する事もできます。それには、元のブロック線図を変更する必要はありません。

変更は、モデルウィンドウ（下図）にある RT ｵﾌﾟｼｮﾝ(O)ボタンでリアルタイムオプションＤＬＧを開きます。

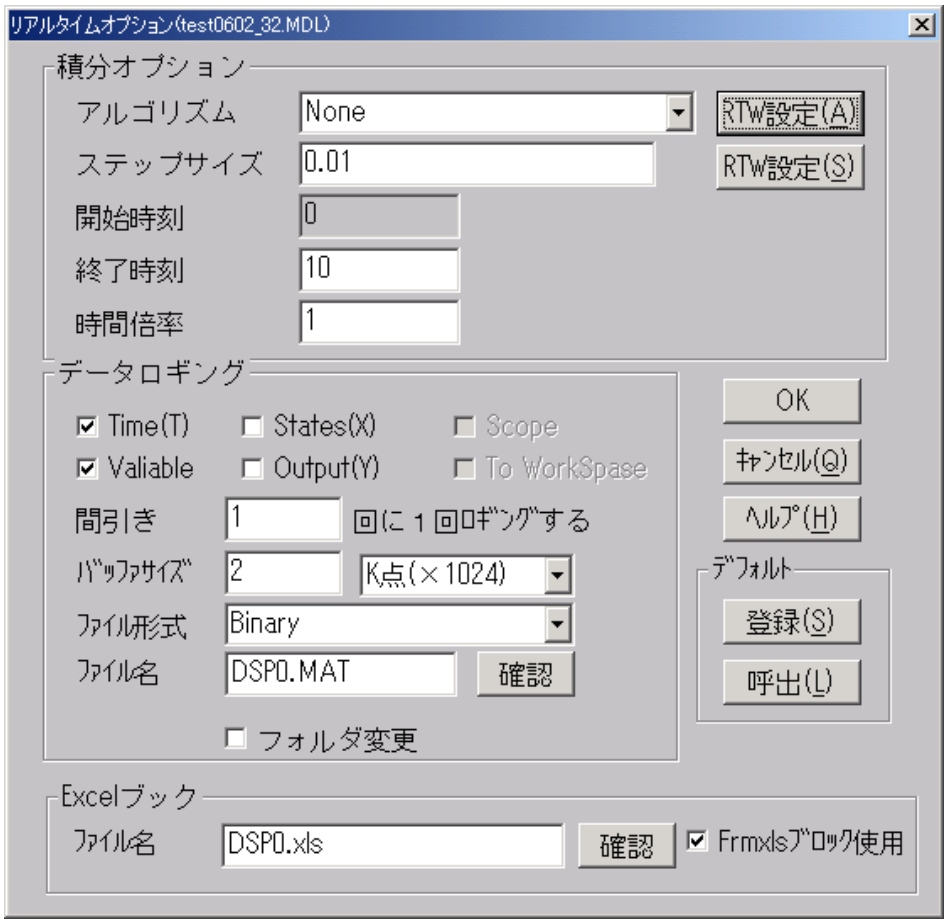


注：この図のアプリケーションの中のチャイルドウィンドウがモデルウィンドウと呼ばれるウィンドウです。この図では、モデルウィンドウを見易くする為、それ以外の表示を隠蔽したうえ、大きさを調整してありますので、通常ＲＴモデルをロードした直後とは表示が異なります。

シングルＤＳＰモデルの場合は使っているボード番号がモデルウィンドウに表示されていますので、そのまま RT ｵﾌﾟｼｮﾝ(O)ボタンを押します。

マルチＤＳＰモデルの場合、設定は個々のＲＴモデル毎にありますので、モデルウィンドウで対象ＤＳＰ番号を選択してうえ、 RT ｵﾌﾟｼｮﾝ(O)ボタンを押します。対象ＤＳＰ番号はその右にあるスライダーで選択します。

１個のＲＴモデルには１個のＲＴオプションしか存在しませんので、シングルＤＳＰモデルとマルチＤＳＰモデルと同じＲＴモデルを使う場合、例えばシングルＤＳＰモデルとしてロードした際の設定を変更すると、マルチＤＳＰモデルの同じＲＴモデルに、その変更が反映されます。



リアルタイムオプションの設定画面です。積分オプションを初めモデル実行に際してのオプションを設定します。

アルゴリズムと、ステップサイズは、その右にある RTW 設定ボタンを押すとＲＴＷでコード生成した時点のブロック線図の設定が呼び出されます。呼び出されるのは、ＲＴＷでコード生成した時点の設定である事に注意してください。ブロック線図の Simulation parameters の設定を変更しても、ＲＴＷでコード生成しないとその変更内容はＲＴモデルには反映されません。

ブロック線図の設定と異なる設定で動作させたい場合、アルゴリズムはコンボボックスのリストの中から使いたいアルゴリズムを選択します。ステップサイズはその時間値［単位：秒］を直接入力します。指数形式（1.5e-003）または、実数形式（0.0015）いずれでも構いません。

開始時刻は、通常は０のままとしてください。

終了時刻は、一定時間ＲＴモデルを動作させたら自動的に停止させたい場合、その時刻値［単位：秒］を、それ以外は充分に大きな値を設定します。

時間倍率は通常は１としてください。

また、モデルにて frmxls ブロックを使用した場合必ず“Frmxls ブロック使用”へチェックし Excel ブックの指定をしてください。ここで指定すると実行前に Excel より DSP へデータ転送が実行されます。ブロックを使用していてこの指定をしない場合モデルの動作は保証できません。
設定が完了したら、ＯＫボタンを押してください。変更内容はディスクファイルに保存され次回ロード時には同じ設定が自動的に再現されます。尚、ここで設定した値はＤＳＰ上でのＲＴモデルの実行に採用されますが、ブロック線図の Simulation parameters には反映されません。

8　実行開始

実行開始は、メニューから 実行（R）～ 実行開始(B)（又はツールバーの実行開始ボタン）で行います。

正常に実行が開始された場合は、ＲＴＭＯＮのステータスバー（場面最下段）に「起動しました」と表示されます。

9　変数モニター

変数とは、ブロック線図に配置されたブロックから出力された信号を意味します。ブロック線図の表示でいうとブロックとブロックの間の結線に相当します。ＤＳＰのＲＴモデルではＤＳＰ内部のメモリーに割り当てられ、これを表示／ロギングする事ができます。変数にはそれを出力するブロックの名称が割り当てられます。

9.1　変数の選択

変数モニターはＲＴＭＯＮのメニューから、表示(V) ～ 変数グラフ表示(V) でモニター変数選択ＤＬＧ（下図）を開き、ここでモニターしたい変数を選択します。

モニター変数選択ＤＬＧには、ロードされたＲＴモデルにある参照可能な変数の一覧が表示されます。例えばブロック線図のトップレベルに Gain ブロックが有る場合、この名称を元にした変数名称として root_Gain と表示されます。ブロック線図上の名称と変数名称との関係については、サブシステム・ビューアーのヘルプを参照してください。

ＲＴモデルを作成後、初めてロードした状態では何も選択されない状態となっていますので、このＤＬＧのリストの中から観測したい変数名をマウス左ボタンでクリックし選択状態にします。例えば、トップレベルの Gain1 と Sum の各ブロックの出力を観測したい場合、root_Gain1 と root_Sum を選択します。（下図）

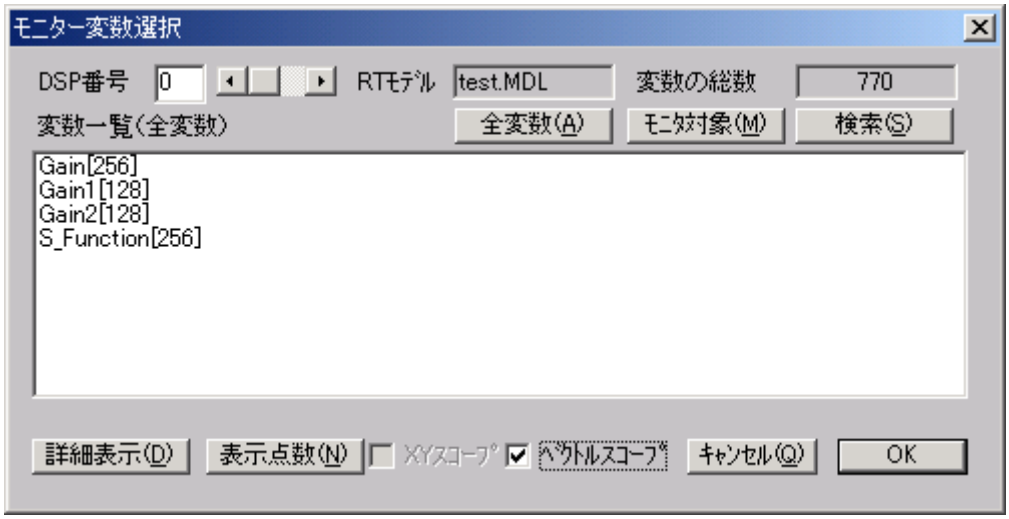
マルチＤＳＰの場合はＤＳＰ番号の右のスライダー又はエディットボックスで目的とするＤＳＰボード番号を選択したうえ、そのボードの変数を選択します。

誤って選択してしまった変数は、再度マウス左ボタンでその変数をクリックすれば非選択状態に戻ります。

選択可能な変数の最大は合計３０個です。

注）観測したい変数名が変数一覧に表示されない場合、弊社ホームページの Q&A「ＭＡＴＬＡＢ対応ライブラリでよくあるＱ＆Ａ」を参照してください。

9.2　ベクトル変数の選択

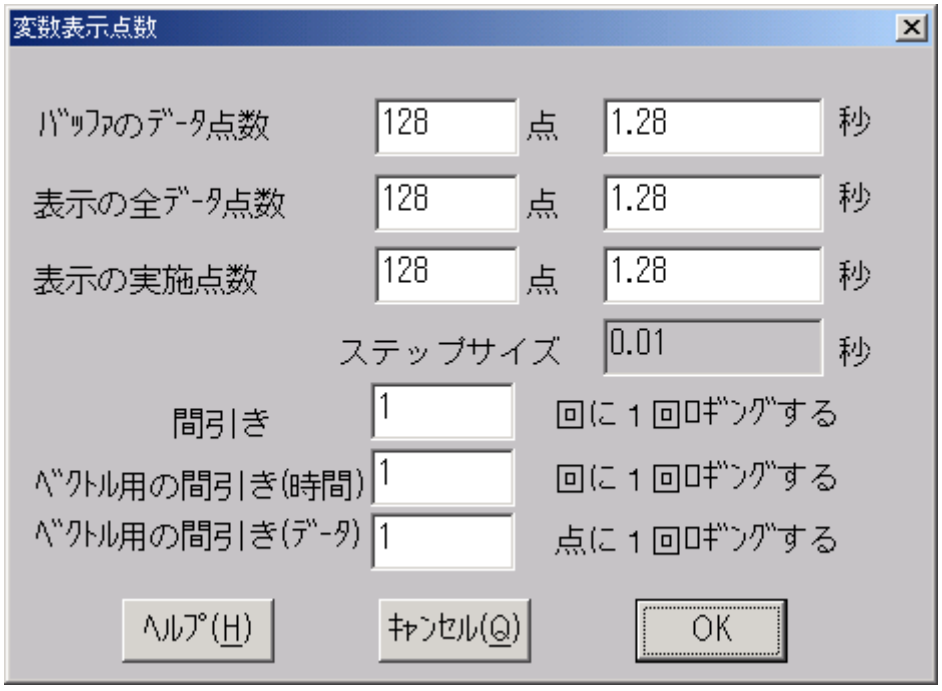


ベクトル情報をもつ変数を観測したい場合は、ベクトルスコープをチェックします。

変数一覧にベクトル情報を持つ変数が表示されるので選択後ＯＫボタンを押してください。

9.3　表示点数の設定

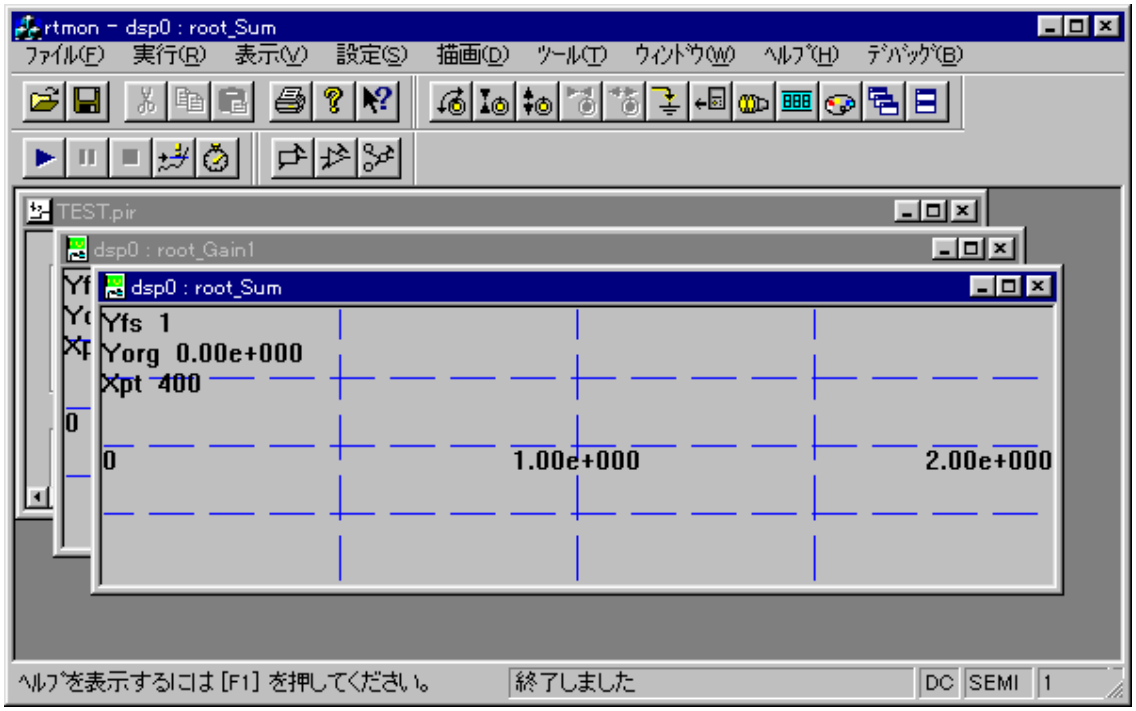
選択が完了したら、表示点数ボタンを押し、変数表示点数ＤＬＧを開きます。



バッファのデータ点数を変更するとその右に表示されている時間が変わります。この時間は、グラフ表示した場合の横軸フルスケール値です。上の例では 1.28 秒となっていますので、グラフ表示の左端から右端までの時間が 1.28 秒となります。これを参照しながらデータ点数を決定します。ただし、間引き設定時は表示時間×間引き数が実際の時間となります。

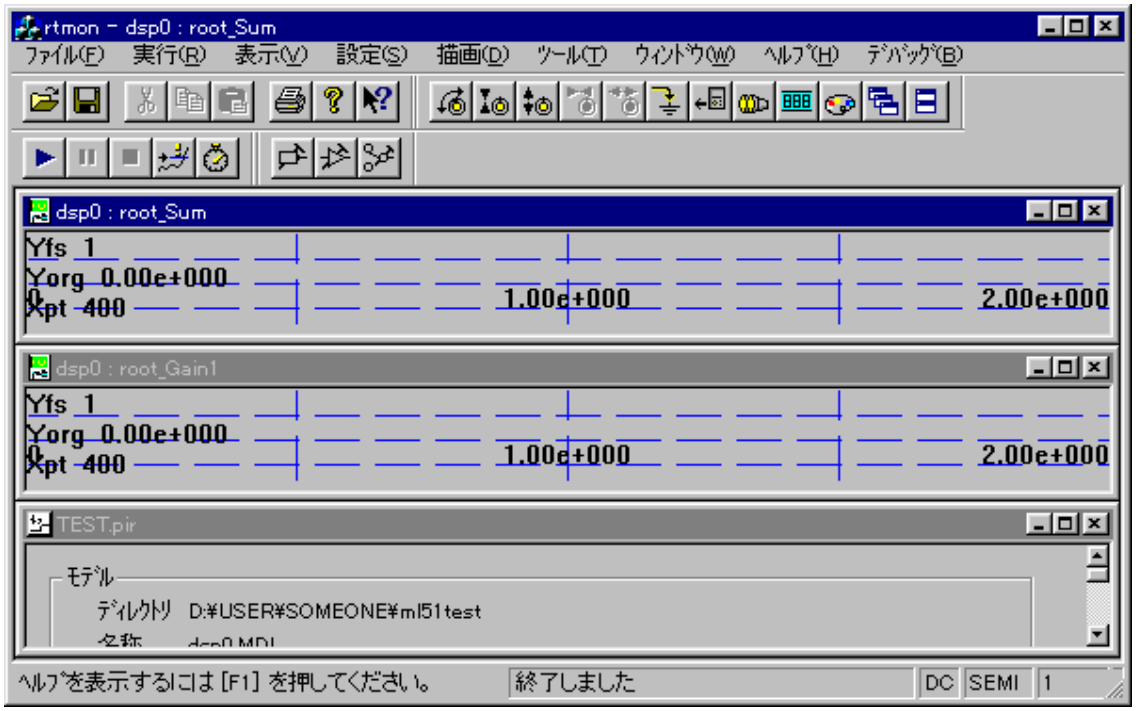
バッファのデータ点数、表示の全データ点数、表示の実施点数は、ここでは共に同じ値に設定します。設定が完了したらＯＫボタンを押してください。

更に、モニター変数選択ＤＬＧのＯＫボタンを押してください。変数選択が完了し、画面には今選択した変数を表示する為のスコープウィンドウが表示されます。



9.4　スコープの自動配置

開いたばかりのスコープウィンドウは上の図の様に、順次重ねて（カスケード）表示されます。マウスで位置・大きさが適宜変更できますが、綺麗に並べて表示するのが目的なら、メニューから、ｳｨﾝﾄﾞｳ(W) ～ 並べて表示(T)（重ねて表示(C)）又は、ツールバーの並べて表示（重ねて表示）ボタンを選択すると、並べて（重ねて）表示するよう自動的に配置されます。また、当面不要な変数はアイコン化したうえで、同操作をすると開いているウィンドウについて再配置されます。下図は“並べて表示”を実施した例です。



9.5　変数の数値読み取り

変数をモニターしているスコープウィンドウ内部をマウス左ボタンで左右にドラッグすると、縦方向の線カーソルが表示され、その位置の、画面左端からのデータ点数値と、変数の数値がステータスバーに表示されます。表示位置はステータスバーの右端ですので、ＲＴＭＯＮのメインウィンドウを小さくしている場合は隠れてしまって見えない場合があります。ＲＴＭＯＮを最大化してもう一度試みてください。

変数がまだスコープに表示されていない場合、ステータスバーの数値を表示する場所には（no data）と表示されます。これによりスコープにデータが無いのか、それとも値が大きすぎて画面からはみ出てしまっているのかが判定できます。

9.6　スコープのレンジ変更

変数の値が大きすぎてスコープ画面からはみ出ている場合、スコープの設定レンジを変更します。

まず、レンジ変更したいスコープウィンドウをマウスでアクティブにします。アクティブか否かは、タイトルバーの色で判断できます。

次に、メニューから、表示(V) ～ 縦軸レンジ拡大(U)（縦軸レンジ縮小(D)）または、ツールバーのＹ軸拡大（Ｙ軸縮小）ボタンを押します。１回押す度に縦軸レンジが約２倍（１／２）に変更されます。縦軸レンジは、１→２→５→１０の様に変更できます。

現在の縦軸のフルスケールは、スコープウィンドウの左端付近にＹｆｓ　１　等として、（及び、ステータスバーにもアクティブなスコープのレンジ値が）表示されています。
１と表示されている場合、縦軸方向の真ん中が０で上端が＋１、下端が－１です。

概ねのレンジ値が予め判っている場合、拡大／縮小ボタンを幾度も押さず、直接レンジ値の設定が可能です。目的とするスコープウィンドウをアクティブにし、メニューから、表示(V) ～ スコープ設定（P）を選択し、スコープ設定ＤＬＧを開きます。

縦軸レンジは“Ｙ軸レンジ設定(FS)”のコンボボックスから選択します。

9.7　Ｙ軸オフセットを付ける

もし、観測しようとする変数に直流分が乗っている場合、「拡大しようとすると画面からはみ出してしまう。かといって、縮小すると波形の細部が見えない」、といった事態¥になります。その場合は、Ｙ軸方向にずらす事ができます。
その方法は３通りあります。
１つ目は、Ｙ軸方向にずらす量（ＤＣオフセット値）を直接設定する事です。設定はＹ軸レンジ設定と同じ、スコープ設定ＤＬＧの“ＤＣオフセット”で行います。例えば、信号の平均値が約０．５の場合、ＤＣオフセットに０．５と入力します。これによりＹ軸方向の真ん中が０．５になります。信号の平均値は、スコープウィンドウをマウス左ボタンでダブルクリックして開いた“スコープ読み取りＤＬＧ”で読み取れます。
２つ目は、マウスを使って視覚的に行う方法です。まず目的の信号が画面に入るまで縮小します。この時、余り小さくし過ぎず、波形がかろうじて画面からはみださない程度とします。この状態でスコープウィンドウの内部をマウス右ボタンで上下にドラッグします。

ドラッグしている間は横方向の線カーソルが表示され、ステータスバーにカーソルの位置の信号値が表示されます。このままドラッグし、波形のＹ軸方向の中心と思しき位置でマウス右ボタンを放します。これにより、ボタンを放した時点のカーソルの位置がＹ軸方向の真ん中に来る様、Ｙ軸方向にグラフが移動されます。これは、ボタンを放した時点のＹ軸値を読み取り、先のＤＣオフセット値に設定したのと同じ意味となります。
３つ目は、自動的に行う方法です。スコープ設定ＤＬＧを開き、入力結合をＡＣに変更します。測定機のオシロスコープのＡＣ／ＤＣ結合のイメージ的には似ていますが、違う点は、実際にハード的にＡＣ結合にするのではなく、データ処理によりＡＣ結合の様な動作をしている点です。この設定の場合スコープは、以下の手順でＤＣオフセットを自動調整します。（１）１回目の表示は現在のＤＣオフセット値でグラフ表示します。（２）表示したデータの平均値を求めます。（３）この値をＤＣオフセット値に設定します。従って２回以上表示を反復する場合に効果が現れ、先回表示したデータの平均値が、画面真ん中に来るよう今回のデータの表示が行われます。平均値が安定している、比較的周期性のある信号の場合に効果が発揮されます。逆に、突発的な現象や、信号変化が緩やかで１周期が横軸の長さより長い場合等は逆効果かも知れません。

9.8　Ｙ軸オフセットの除去

Ｙ軸方向に移動したグラフを元に戻し、Ｙ軸方向真ん中を数値０にするには、スコープ設定ＤＬＧでＤＣオフセット値を０に設定するか、又はメニューから、表示（Ｖ）～ DC ｵﾌｾｯﾄ除去(Z)を選択するか、又は、ツールバーの表示ｵﾌｾｯﾄ除去ボタンを押します。

9.9　変数の削除

不要になった変数モニターを削除するには２通りの方法があります。
１つ目の方法は、不要なスコープウィンドウ右上のＸボタンを押す事です。
もう１つの方法は、モニター変数選択ＤＬＧを開き、不要となった変数をマウスで非選択状態にしたうえ、ＯＫボタンを押す事です。

9.10 スコープ状態の保存・選択

現在のスコープ状態が保存できます。通常、初めてモデルロード時は何も表示されません。２回目以降のロード時は前回終了時のスコープ状態がモデル名.scp というファイル名で保存されているのでこのファイルより状態を復帰します。このファイルとは別に現在のスコープ状態を保存・選択できます。

保存･･･　表示→スコープ保存　　ファイル名をつけて保存してください
選択･･･　表示→スコープ選択　　保存したファイル名にて呼び出してください。

10　計算所要時間の測定

ＤＳＰ内部でブロック線図の計算にどれだけ時間が消費されているかを知る事ができます。原理的にステップサイズはこの時間より短くする事は出来ませんから、この時間を知る事は、ステップサイズがどこまで短くできるかを知る上で重要な事です。又、時間余裕を知る意味でも重要な事です。

まず、正確な計算所要時間を把握するには、ステップサイズを充分大きくしたうえで、ＲＴモデルを動作させる必要があります。計算所要時間がステップより長いと異常と見なしてＲＴモデルが自動的に停止する仕組みになっている為です。
ＲＴオプション設定のステップサイズを、例えば０．１秒とか１秒等、充分大きい値に設定し、モデルを起動します。

次に、メニューから 表示(V) ～ 計算所要時間表示(N) （又はツールバーの計算所要時間ボタン）を選択し、計算所要時間ＤＬＧを表示します。
尚、ここで計算所要時間が表示できるモデルはＲＴＷで生成したモデルのみです。

この表示の更新の間隔は約１秒ですが、ステップサイズがこれより長いと（例えば１０秒）、正しい所要時間値が表示されるのは、ＲＴモデル起動、又はこのＤＬＧを開いてからステップサイズ（１０秒）経過後となります。

通常、表示値は一定ではありません。ばらつきます。一般のＲＴモデルの場合、ばらつきが数μ秒以内であれば正常と見なして構いません。
しかし、ＲＴモデルがユーザ定義関数等を含んでいて、この関数の実行時間が回を追う毎に変化する場合は、それがこのＤＬＧに反映され、表示値がばらつきます。

また、積分アルゴリズム、データロギングの有無、変数モニタの有無、イネーブルドサブシステムやトリガードサブシステムが有る場合はその実行の有無により、所要時間は大きく変わります。これらの条件を色々変え、最長の値を見極める必要があります。
ＲＴモデルを安全に動作させるには、数値を何回か測定し、その最高値を所要時間と見なし、ステップサイズはこれより長い時間を設定しないといけません。
上記のような所要時間にばらつきがでるモデルについては、“ﾋﾟｰｸ”欄へチェックをして計測してください。実行時からの最長の時間が表示されます。

11　データロギング

データロギングは、ＤＳＰ内部メモリーにＲＴモデルの動作中のデータを貯え、ＲＴモデルの動作終了後これを取り出しファイル化するメカニズムです。

ロギングデータは、１ステップの計算を行う都度１組みのデータが生成されます。従ってステップサイズＴでＮ秒間動作を継続するとデータはＮ／Ｔ組み生成されます。一方、ＤＳＰに保存できるデータの量にはメモリー容量の都合で上限があります。これを超える事は出来ません。保存しきれないデータは保存領域がいっぱいになった段階で、古い順に捨てられます。

11.1　ロギング項目

ロギングできる項目は以下の通りです。

○現在時刻値	Time(T)	
○状態変数	States(X)	
○出力変数	Outputs(Y)	
○変数	Valiable	
○Scope ブロック	Scope	（ＭＬ4.2c のみ）
○To Workspace ブロック	To WorkSpace	（ＭＬ4.2c のみ）

現在時刻値は、ブロック線図に Clock ブロックを付けた場合、このブロックの出力値と同じです。後にＭＬで時刻データが必要になる場合、わざわざブロック線図に Clock ブロックを取り付けなくても、現在時刻値が保存できます。

状態変数は、モデルが状態変数を持っている場合、計算ステップ毎の状態変数値が保存できます。但し、ここで保存できるのは、メジャータイムステップ（ステップサイズ間隔）のみです。

出力変数はブロック線図のトップレベルに取り付けた出力ポートに出力されるデータです。これは、ブロック線図から大域変数を経由してユーザ定義関数へ渡されるデータでもあります。

変数は全変数のうち、変数モニターで選択した変数についてロギングされます。

Scope ブロック／To Workspace ブロックは、ＭＬ4.2c の場合にみ有効です。各ブロックへ出力されるデータがロギングされます。

11.2　ロギング項目選択

上記データロギング項目のどれを取り込むかは、リアルタイムオプションＤＬＧの下半分、データロギンググループのチェックボックスで設定します。

上の例では、現在時刻、状態変数、変数、が選択されています。

11.3　データ間引き

長時間に渡りデータを保存したいが、全データを保存するにはメモリー容量が不足する場合、データの間引きを使います。

間引き設定は、１を設定した場合、毎回データが保存されます。２では２回に１回、３では３回に１回のデータが保存され、それ以外のデータは破棄されます。

アンチエリアシング等の処理は行われていません。必要なら、ブロック線図上でフィルターリングを行う必要があります。

11.4 データ点数

データ点数は、ＤＳＰ内部に確保するバッファの長さと考える事ができます。点数の単位は２種類から選択できます。上の図でバッファサイズの右の、点と表示されているコンボボックスを開きます。選択肢は 点 と K 点(×1024) です。点の場合は、その左に表示されている数値そのものがバッファの点数＝保存できるデータ点数となります。K点(×1024)の場合は、その左の数字×1024 点がバッファの点数となります。例えば、２K点と 2048 点とは同じ意味を持ちます。

このデータ点数は、ロギング項目毎、変数毎にこの点数分用意されます。

11.5 ファイル形式・ファイル名

ロギングデータをファイルの保存する場合の形式と名称を設定します。

形式は通常は Binaly としてください。この形式がＭＬのデータファイルと互換があります。ASCII はＭＬ以外のプログラムとデータ交換する場合用に使用します。ＭＬでは特殊な場合を除きロード出来ません。

ファイル名の拡張子は、Binaly の場合は .MAT 、ASCII の場合は .DAT としてください。

また、“フォルダ変更”にチェック無し時は モデル名_grt_rtw フォルダへ、チェック有り時はモデルのあるフォルダへ保存されます。（MATLAB Version 6.0 以降）

11.6 マニュアルトリガー

ロギングデータは、モデル実行時に任意の時刻に取得することができます。

ﾄﾘｶﾞｰON

をｸﾘｯｸするとその時点でデータロギングが停止します。ロギングを再スタートするにはﾄﾘｶﾞｰ有効

をｸﾘｯｸしてください。

12　モデルの終了

　ＲＴモデルは次の条件で終了します。

○終了時刻に達した時。

○ＲＴモデルで Stop simulation ブロックが使われていて、停止条件が成立した時。

○ＲＴＭＯＮで停止操作をした時。

○トリガー機能が有効でかつ自動停止が有効となっており、トリガー条件が成立した時。

ＲＴＭＯＮでは、以下の何れかの操作を行うとＲＴモデルの動作を停止させる事ができます。

１、メニューから、実行(R) ～ 実行終了(E)を選択する。

２、ツールバーの実行終了ボタンを押す。

３、モデルウィンドウのＸボタンを押す。

４、別のＲＴモデルをロードする。

できるだけ１又は２の方法を使用してください。

13　ロギングデータの保存

ＲＴモデルでロギングしたデータは、ＲＴモデルの停止後ファイルに保存します。

13.1　シングルモデルの場合

シングルモデルの場合、上書き保存（予め決めておいた名称で保存）と新規保存（保存の時点で名称を決める）ができます。

13.1.1　上書き保存

予め名称を決めておくには、リアルタイムオプションＤＬＧのファイル名エディットボックスで決定します。

上書き保存の実行は、メニューから、ファイル(F) ～ データ上書保存(S)、又は、で保存します。保存されるデータは、ＲＴモデルが格納されているフォルダーに保存されます。

13.1.2　新規保存

上書き保存の実行は、メニューから、ファイル(F) ～ データ新規保存(A)を選択します。

ロギングデータファイル新規保存ＤＬＧが開きますので、ここで既存のファイル名称を確認のうえ、ファイル名(N) に保存したいファイル名称を入力し、保存(S)ボタンを押します。

新規保存ですが、既存ファイル名を指定した場合は警告を表示したうえ、上書き保存されます。

13.2　マルチモデルの場合

マルチモデルの場合、上書き保存（予め決めておいた名称での保存）しか行えません。

データファイルはＤＳＰ毎に１個ずつ個別に保存されます。これは、複数のＲＴモデルで同じ名称の変数（＝別々のブロック線図に同じ名称のブロック）が存在する可能性がある為、複数のロギングデータを１つにまとめられないことによります。

個々のデータファイルの名称は、リアルタイムオプションＤＬＧで予め設定しておきます。

上書き保存の実行は、メニューから、ファイル(F) ～ データ上書保存(S)、又は、で保存します。保存されるデータは、ＲＴモデルが格納されているフォルダーに保存されます。

14　ロギングデータのＭＬでの利用

ＲＴＭＯＮで保存したデータをＭＬで利用するには、MATLAB Command Window で load コマンドでロードします。これで初めてＤＳＰで採ったデータがＭＬワークスペースへロードされます。

マルチモデルのロギングデータの場合、各ＤＳＰのデータは個別のファイルになっていますので、load コマンドを反復してロードします。この時注意が必要なのは、各データで同一名称が存在する場合、警告されず、単に後でロードされたデータによって置き換えられる事です。

ロードされたデータの変数名は、以下の通りです。

現在時刻	T
状態変数	X
出力変数	Y
変数	変数選択ＤＬＧに表示された名称と同じ。
Scope	Scope の名称
To Workspace	To Workspace ブロックに設定した名称

Ｔはバッファサイズ×１の縦長ベクトルです。
Ｘはバッファサイズ×状態変数の数のマトリクスです。列ベクトルが１つの状態変数の時系列ベクトルです。行方向にどんな状態変数が格納されているかは、現状では簡単に表示する機構が用意されていません。ＲＴWが出力したソースファイル *model*.h の構造体宣言

```
typedef struct States_tag {
・・・・・・・・
} States ;
```

が参考となります。データはこの構造体のメンバーの順に並んでいます。メンバーが配列の場合はその添え字の０から順に並んでいます。
Ｘのなかの特定の状態変数例えば２行目の状態変数の時系列を取り出し描画するには、MATLAB Copmmand Windowd で

```
X2 = X(:,2); plot(T,X2);
```

と実行します。
Ｙはバッファサイズ×出力点数のマトリクスです。列ベクトルが１つの出力の時系列ベクトルです。Ｘ同様に操作できます。
それ以外のデータも全て列ベクトルです。

15　自動テスト用ブック作成

メニューの設定→自動テスト用ブック作成　にて作成できます。このブックは Microsoft 社製 Excel で作成するブックのことです。予め PC に Excel がインストールされていないと使用できません。動作可能 Excel のバージョンについては、“自動テストユーザーズ・マニュアル”を参照してください。

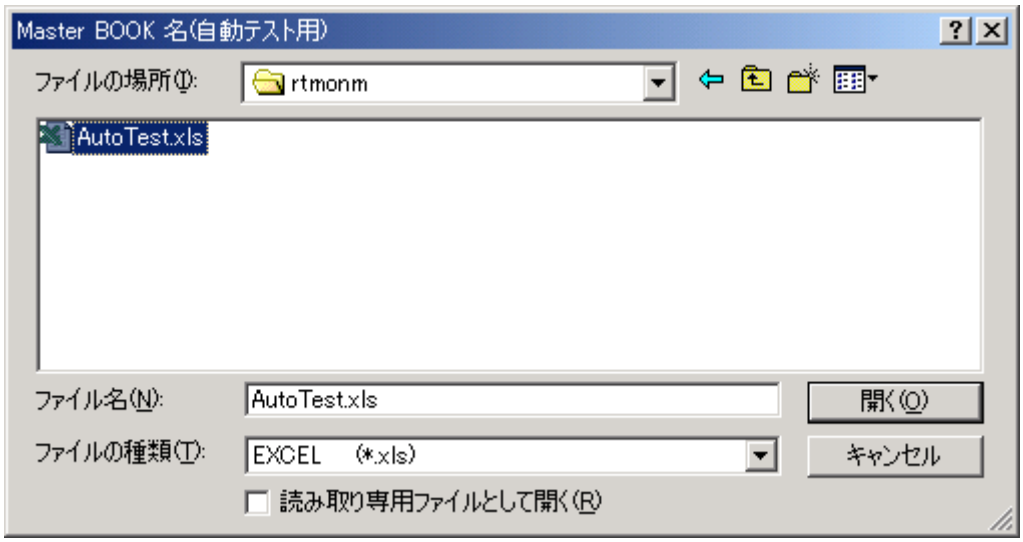
このブックは、自動テスト実行時における実行条件と、Frmxls/Toxls ブロックで使用するデータの場所を示すものです。自動テスト実行時又は RTMON にて Frmxls ブロック付のモデル実行時には必ず作成してください。

尚、Frmxls/Toxls ブロック実行時に、Excel で指定した内容とブロックで指定した各パラメータの内容の整合性チェックはしませんので、ブック作成時に確認してください。もし整合が取れていない場合は、モデルの動作は保証できませんので、予めご了承ください。

整合性とは･･･List シートにて指定する Index とブロックで指定した Index 値の一致、List シートで指定するブロックの種類などです。詳細は“I/O デバイスライブラリ”又は“自動テストユーザーズ・マニュアル”を参照してください。

15.1　基本ブック指定

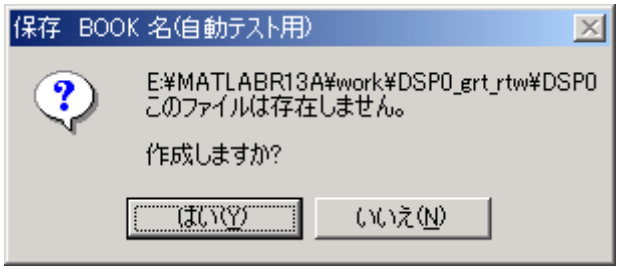
モデルをロードします。次に自動テスト用ブック作成を選択し、以下のような基本となるブック “MATLAB ｲﾝｽﾄｰﾙﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥rt¥Autotest.xls”を指定します。



15.2 作成ブック指定

作成するブックの保存場所、名称を指定します。

ブックを新規作成時は以下のダイアログが表示されますので“はい”を選択してブックを作成します。



以上で現在の RT オプションの状態でブックが作成されますので、“自動テストユーザーズ・マニュアル”を参照の上実行条件、データを指定しブックを完成し、自動テストを起動してください。

＜＜　メ　モ　＞＞

ＭＡＴＬＡＢ対応ライブラリ（マルチ）
ＡＤＳＰ３２４－４１Ｂ
ＡＤＳＰ４０４－２４１Ｂ
ＡＤＳＰ６７４－３４１
リアルタイムモニター・チュートリアル

第１版　　1998.08.31
第１１版　　2006.02.08

中部電機株式会社 ＣＳ事業部
TEL　0532 – 63 – 5311
FAX　0532 – 61 – 9566

自動テスト

ユーザーズガイド

中部電機株式会社

目　次

I.　概要 1

II.　動作環境 1

III.　実行手順 2

1.　モデル作成 2
2.　パラメータファイル作成 3
3.　ＥＸＣＥＬＢＯＯＫ作成 4
4.　自動テスト起動 5

IV.　実行サンプル 7

1.　モデル作成　→　モデルビルド 7
2.　パラメータファイル作成　→　ＲＴＭＯＮにて作成 7
3.　ＥＸＣＥＬＢＯＯＫ作成　→　テンプレートブックをコピーして作成 8
4.　起動　→　ＭＡＴＬＡＢ　ＣＯＭＭＡＮＤ　ＷＩＮＤＯＷから起動 9

V.　エラーコード表 １０

I. 概要

自動テスト機能は、ＭＡＴＬＡＢ Ｃｏｍｍａｎｄ Ｗｉｎｄｏｗと弊社製リアルタイムモニタ（以下ＲＴMON）より起動できる、リアルタイムテストを自動化するソフトウェアです。実行時に、ステップサイズ／終了時間等の動作条件や、Ｇａｉｎ／Ｃｏｎｓｔａｎｔ／Ｓｗｉｔｃｈ等のパラメータをダイナッミックに変更できます。

また、新たに自動テスト用Ｉ／Ｏブロックとして frmxls / toxls が追加されたことにより、入力データを予め他の計測器等でＥｘｃｅｌファイルとして準備し、実行時にダイナミックに入力できたり、出力データをＥｘｃｅｌファイルとして保存も可能となりました。

このことにより、ＭＡＴＬＡＢスクリプトにて起動コマンドを記述すれば、テスト毎に条件・パラメータを最適化しながら連続テストも可能です。

II. 動作環境

- ＭＡＴＬＡＢ対応ライブラリ（Ver ５.１.０以上）が動作する環境
  （動作環境は別紙ＭＡＴＬＡＢ対応ライブラリユーザーズマニュアルを参照してください。）
- Microsoft Excel ２０００ for Windows

＊ Microsoft は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。

＊ ＭＡＴＬＡＢは、米国 The Math Works Inc.の登録商標です。

III. 実行手順

1. モデル作成

ＭＡＴＬＡＢにてモデルを作成し、ビルドします。

自動テスト用Ｉ／Ｏブロックとして、frmxls / toxls ブロックが使用できます。

（frmxls / toxls ブロックは自動テスト時のみ有効のため、それ以外で実行時は動作が保証されません。）

frmxls…Excel 形式の時刻付データを実行時にダイナミックに入力するブロック。

＜Parameter＞

・VariableIndex … データ ID として０～９９までの整数。
toxls の VariableIndex と重複しないように設定。

・SampleTime … 入力サンプルタイム。

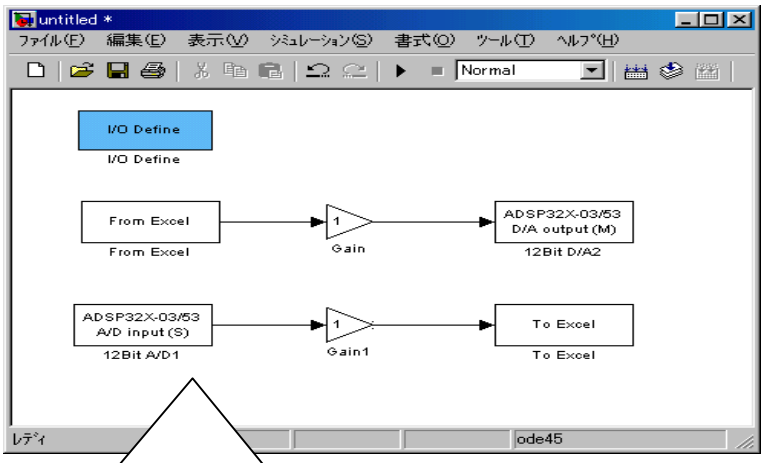
toxls …実行結果を Excel 形式の時刻付データとして出力するブロック。

＜Parameter＞

・VariableIndex … データ ID として０～９９までの整数。
frmxls の VariableIndex と重複しないように設定。

・Decimation … 間引き数指定。（ex. ２指定時サンプルタイムの２回に１回出力）

・SampleTime … 入力サンプルタイム。



toxls と frmxls ブロックの VariableIndex は、重複しないように設定してください！

自動テスト～２

2. パラメータファイル作成

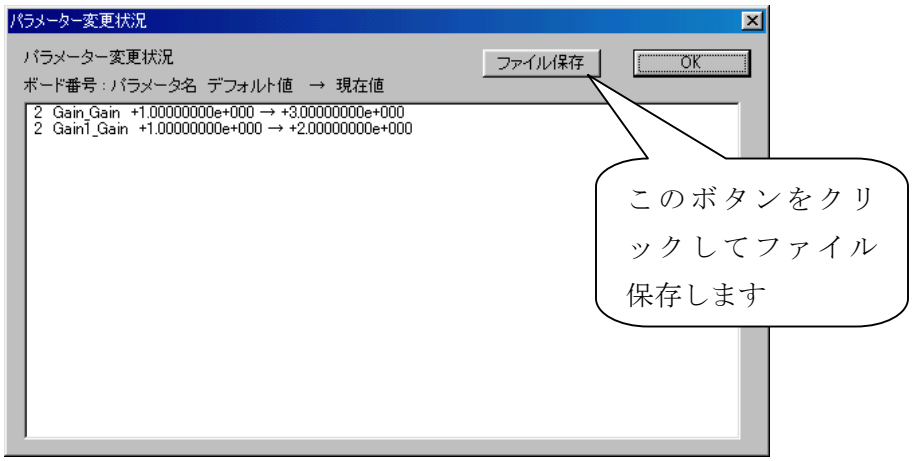
① ＲＴMONを起動し、メニュー → ＲＴモデルロード を選択モデルをロードしてください。

② A）メニュー → 設定 より設定したい項目（ゲイン・定数・閾値・その他のパラメータ）を選択し値を設定してください。

B）弊社製モデルコントロール（別売オプション）の、ボリューム・スライダ・数値入力コントロールからもブロックパラメータの設定が可能です。

③②にてすべて設定が終了したら、 メニュー → 設定 → 変更状況の表示 を選択し“ファイル保存“にてＴＥＸＴファイルに保存します。

＊その他のパラメータについては、一部実行時に変更できないものがあります。

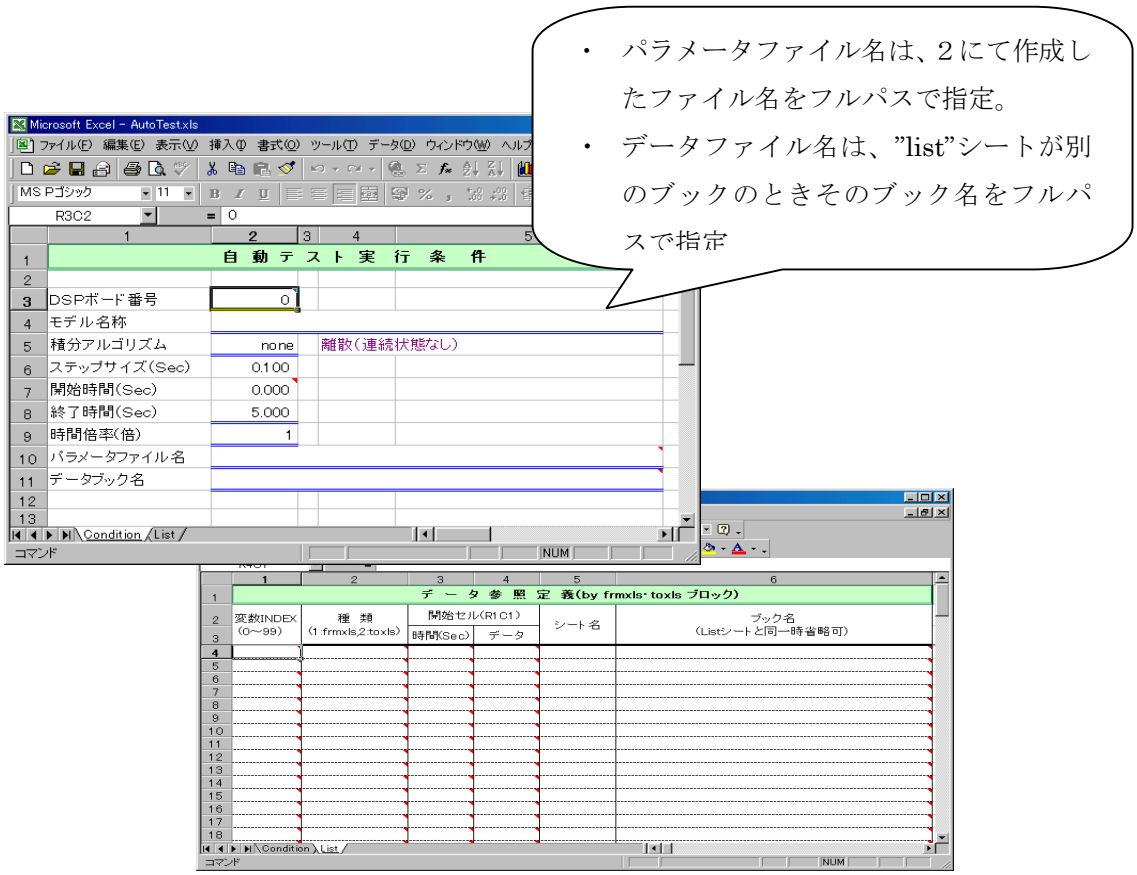


3. ＥｘｃｅｌＢｏｏｋ作成

下記ディレクトリにテンプレートＥｘｃｅｌＢｏｏｋ“Autotest.xls”が格納されていますので、このＢｏｏｋを利用し動作条件を設定してください。このＢｏｏｋは、“condition”“list”の２種類のシートから構成されています。
また、Ｂｏｏｋの作成はＲＴＭＯＮからも可能です。（メニュー→設定→自動テスト用Ｂｏｏｋ作成）

“condition”…動作条件設定用
“list”　　…frmxls / toxls ブロックのデータ設定用。frmxls / toxls ブロック未使用時もこのシートは必要です。

＜テンプレートＥｘｃｅｌＢｏｏｋディレクトリ＞
MATLAB ｲﾝｽﾄｰﾙﾃﾞｨﾚｸﾄﾘ¥rtw¥c¥adsp32x¥rtmonm　　（adsp324 時 ）
MATLAB ｲﾝｽﾄｰﾙﾃﾞｨﾚｸﾄﾘ¥rtw¥c¥adsp32x¥rtmonm67　（adsp674 時 ）



4. 自動テスト起動

1) MATLAB Command Window 利用（MATLAB　Ver5.2.1 以上対応）

＜コマンド＞

- ｓｅｔｏｐｔｎ　…Condition シート：動作条件設定
- ｓｅｔｐｒｍ　…Condition シート：パラメータファイル名設定
  （ＲＴＭＯＮにて作成したファイルをフルパス指定）
- ｓｅｔｄａｔａ　…Condition シート：データブック名設定
  （“list”シートのブック名が”Condition”シートのブックと異なるときのみプルパス指定）
- ａｕｔｏｔｅｓｔ…自動テスト起動
- ｓａｍｐｌｅ　…自動テスト連続起動

上記コマンドを利用して実行します。各コマンドとも終了時には終了メッセージがエラー時には、エラーメッセージとエラーコードが表示されます。エラーの詳細についてはエラーコード表を参照してください。
また、コマンドの引数詳細は、ヘルプを参照してください。

例　help setoptn

◎コマンド使用例

①>>setoptn(‘autotest.xls’,0.01)　← **step size 0.01Sec 設定**

②>>autotest(‘autotest.xls’)　← **自動テスト起動**

AutoTest start!<C:¥MATLABR132¥work¥test10_grt_rtw¥autotest.xls>
--> Finish!

↑　**起動、終了メッセージ表示**

④>>setoptn(‘autotest.xls’)　← **現在の設定値表示**

⑤>>setprm(‘autotest.xls’, ‘’)　← **パラメータファイル名消去**

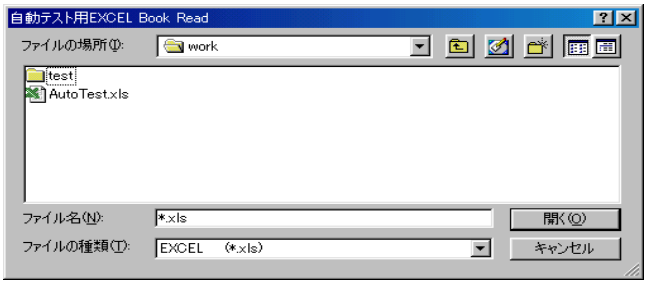
⑥>>sample(‘autotest.xls’,’autotest1.xls’)

↑ **自動テスト連続実行**

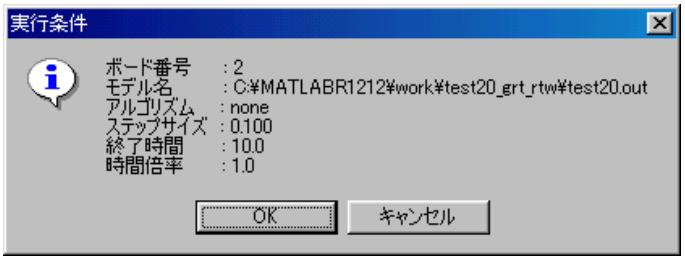
2)RTMON 利用

①ＲＴＭＯＮを起動し、メニュー → 実行 → 自動テスト を選択します。
（モデル未ロード時のみ有効）

②下記ダイアログにて計画用Ｅｘｃｅｌ Ｂｏｏｋを選択します。



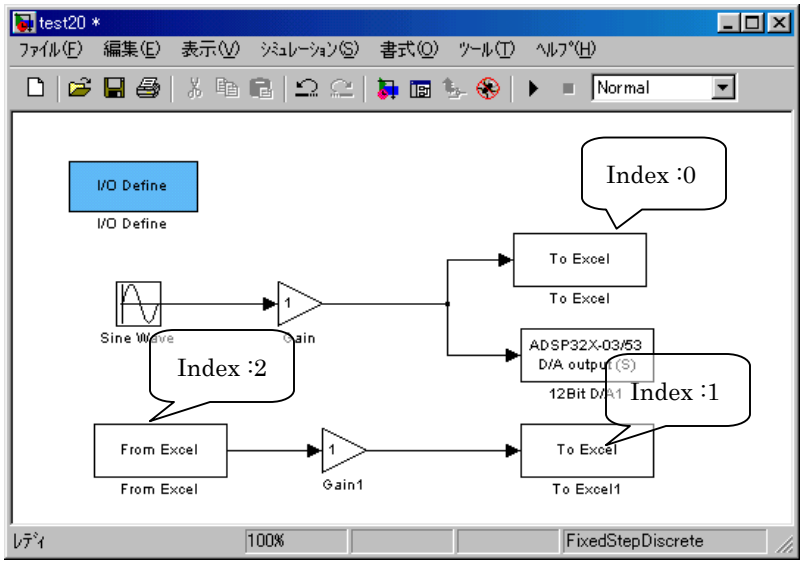
③「開く」を選択すると実行条件確認メッセージが表示されます。内容を確認し応答してください。「ＯＫ」を選択するとテストが開始されます。
実行中は、実行中ダイアログが表示され、終了するとダイアログが消えます。
実行中に関係するＥｘｃｅｌＢｏｏｋを操作（開く・編集・消去など）するとエラーとなります。



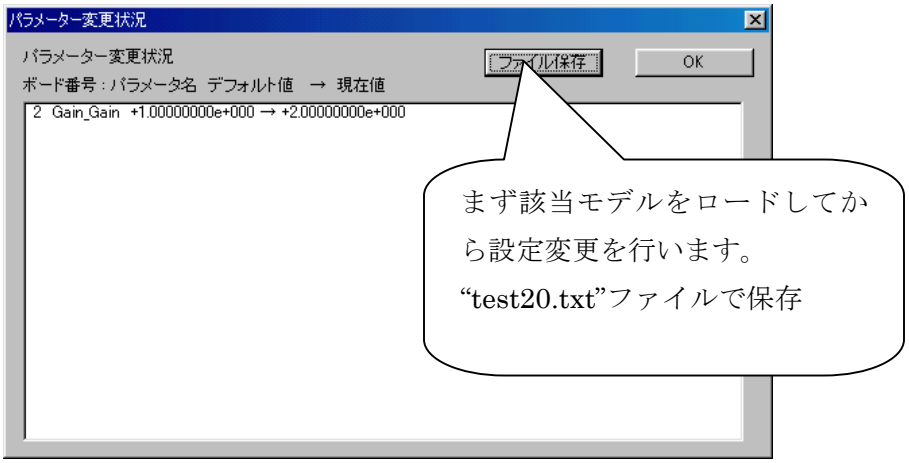
④テスト実行中にエラーが発生すると下記ダイアログメッセージが表示されます。詳細については、エラーコード表を参照してください。

Ⅳ. 実行サンプル

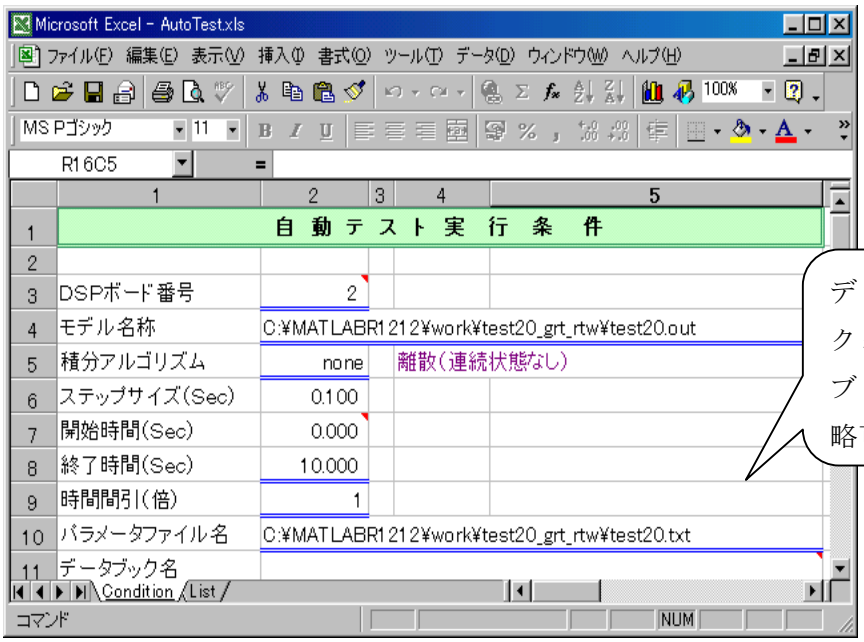
1. モデル作成　→　モデルビルド



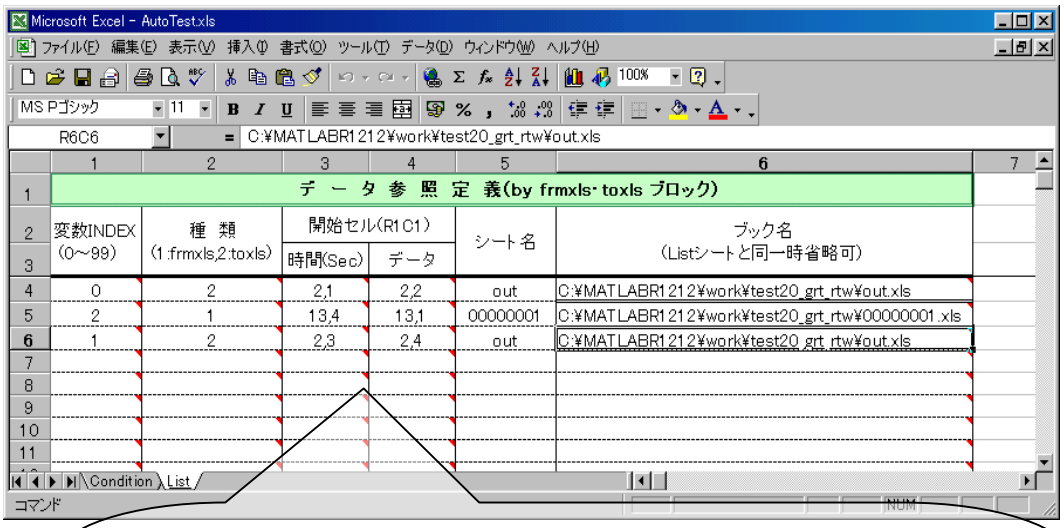
2. パラメータファイル作成　→　ＲＴＭＯＮにて作成



3. ＥｘｃｅｌＢｏｏｋ作成　→　テンプレートブックをコピーして作成

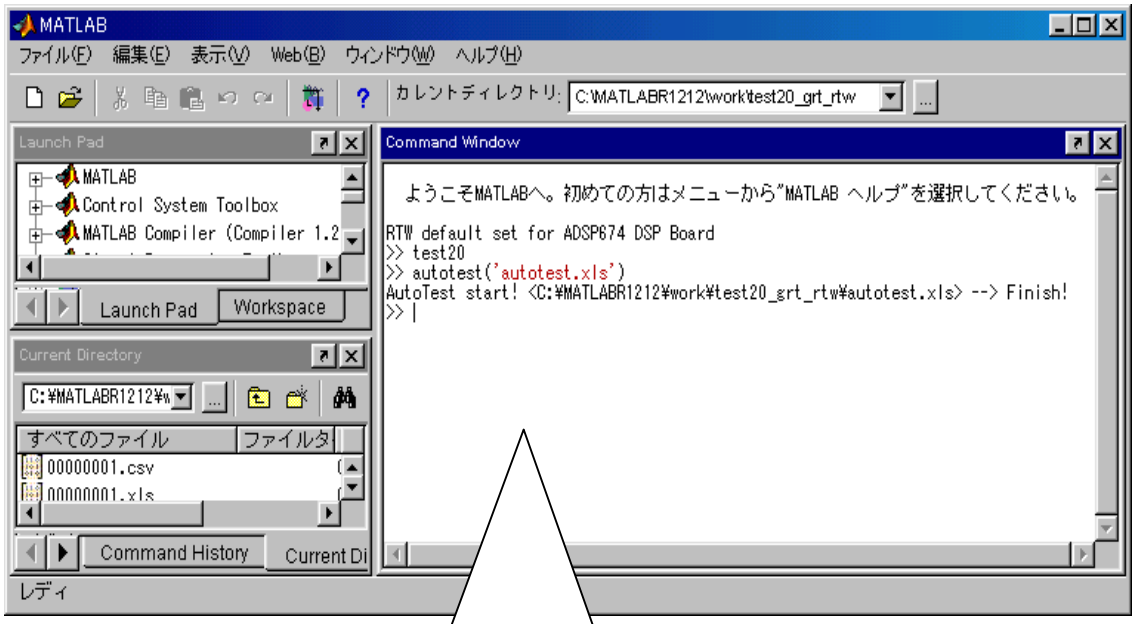


データブック名は同一ブック時省略可。



モデルの Variable Index に対応するように情報を設定します。
セル番地は、R1C1 形式にてコンマで区切り入力。
シート・ブックは、toxls ブロックに限り存在しなければ新規で作成されます。
ブック名は、フルパス名で設定します。但し"list"シートと同一のブックの場合は省略可。

4. 起動 → ＭＡＴＬＡＢ Ｃｏｍｍａｎｄ Ｗｉｎｄｏｗから起動



>>autotest(‘autotest.xls’)
と入力し起動。
カレントフォルダにＥｘｃｅｌＢｏｏｋが
存在する場合、フォルダ名は省略可。

V. エラーコード表

エラー番号	内　　容
１０	AutoTest への引数エラー
２０	Excel アプリケーション起動エラー
３０	IndicationBook Open エラー
３１	IndicationBook ボード番号エラー
３２	IndicationBook モデルエラー
３３	IndicationBook アルゴリズムエラー
３４	IndicationBook ステップサイズエラー
３５	IndicationBook 終了時間エラー
３６	IndicationBook 時間倍率エラー
３７	IndicationBook データ Book エラー
３８	ＤＳＰ初期化エラー
３９	データ Book Open エラー
４０	データ Book Block なし
４１	パラメータファイルエラー
４２	データ転送エラー(frmxls ブロック)
４３	ＤＳＰエラー
４４	ＤＳＰタイムエラー
４５	ＤＳＰその他エラー

本マニュアルの内容は製品の改良のため予告なしに
変更されることがありますので、ご了承ください。

自動テスト for MATLAB 対応 Lib

第 1 版　2003.4.20

中部電機株式会社　CS 事業部
TEL 0532-61-9566
E-Mail csg@chubu-el.co.jp
http://www.chubu-el.co.jp